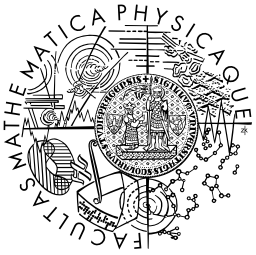


MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Robin Drahovský

Vyhledávání tras v mapách

Informatický ústav Univerzity Karlovy

Vedoucí bakalářské práce: RNDr. Ondřej Pangrác, Ph.D.

Studijní program: Informatika

Studijní obor: Obecná informatika

Praha 2017

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Veľmi rád by som poďakoval RNDr. Ondřejovi Pangrácovi, Ph.D. za vedenie tejto práce, ochotu a odbornú pomoc. Ďalej by som rád poďakoval svojej rodine za podporu a dôveru pri celom štúdiu na vysokej škole, a taktiež každému, kto si túto prácu prečítal a poskytol mi konštruktívnu kritiku.

Název práce: Vyhledávání tras v mapách

Autor: Robin Drahovský

Katedra: Informatický ústav Univerzity Karlovy

Vedoucí bakalářské práce: RNDr. Ondřej Pangrác, Ph.D., Informatický ústav Univerzity Karlovy

Abstrakt: Táto práca popisuje návrh a implementáciu aplikácie, ktorá dokáže vyhľadávať najkratšie cesty v reálnych mapových podkladoch a zároveň berie do úvahy výškové dáta. Vyhľadávanie je rýchle a dá sa prispôsobiť potrebám užívateľa bez nutnosti znova vytvoriť graf, čo túto aplikáciu odlišuje od väčšiny mapových aplikácií, ktoré povoľujú užívateľovi pracovať iba s predpripravenými profilmi vyhľadávania.

Okrem vyhľadávania aplikácia umožňuje vygenerovať vysoko kompaktný graf z mapových dát v OSM formáte, ktorý umožňuje rýchly prístup k informáciám potrebným na hľadanie najkratšej cesty. Aplikácia taktiež dokáže vykresliť mapové dáta, s vlastným štýlom vykresľovania a kontúrami, ale vie aj zobraziť mapu zo serverov projektu OpenStreetMaps.

Klíčová slova: mapy, grafy, grafové algoritmy, nejkratší cesta, A* algoritmus

Title: Searching trails in maps

Author: Robin Drahovský

Department: Computer Science Institute of Charles University

Supervisor: RNDr. Ondřej Pangrác, Ph.D., Computer Science Institute of Charles University

Abstract: This thesis describes design and implementation of an application capable of finding the shortest paths in real world map data, while taking elevation data into account. Path finding is fast and can be adjusted to user's needs without the need for graph rebuilding, what differentiates this application from others, that only allow user to work with predefined search profiles.

In addition to path finding, the application can generate highly compact graph from map data in OSM format. This allows fast access to information necessary for finding the shortest path. The application can render map data with custom rendering style and contour lines, but also can display map from OpenStreetMaps project servers.

Keywords: maps, graphs, graph algorithms, shortest path, A* algorithm

Obsah

Úvod

Hlavnou úlohou tejto práce je navrhnúť a implementovať dátové štruktúry a algoritmy, pomocou ktorých bude možné v reálnom čase vyhľadávať najkratšiu cestu, berúcu do úvahy výškové dáta, v turistických mapách väčších rozmerov.

Druhou, vedľajšou úlohou je vytvoriť aplikáciu, ktorá dokáže vykresliť mapové dáta a na základe užívateľom zadaných parametrov nájsť a zobraziť najlepšiu cestu.

Táto práca sa zameriava na mapové dáta Českej republiky najmä pre ich adekvátnu rozlohu a členitosť terénu. Navyše sa predpokladá, že väčšina užívateľov sa vie v tejto oblasti orientovať.

0.1 Vyhľadávanie najlepšej cesty

Táto práca sa zameriava na riešenie problému vyhľadávania najkratšej cesty medzi dvoma bodmi ohodnoteného, neorientovaného, rozsiahleho a relatívne riedkeho grafu.

V nasledujúcom texte sa pod pojmom graf rozumie implicitne neorientovaný graf.

Definícia 1 (Neorientovaný graf)**.** *Neorientovaný graf G je usporiadaná množina $G = (V,E)$, kde V je neprázdna množina vrcholov grafu G a E je množina neusporiadaných dvojíc (u,v), kde $u \neq v$ a $u,v \in V$. Jednotlivé dvojice z množiny E sa nazývajú hrany grafu G. (Matoušek a Nešetřil, 2009)*

Mapa a graf

Prevod mapy na graf nie je až taký priamočiary ako by sa mohlo na prvý pohľad zdať. Mapové dáta neobsahujú len informácie o cestách, ale zväčša popisujú aj iné objekty, napríklad lesy alebo budovy. To znamená, že pri prevode na graf treba vyextrahovať z mapových dát informácie o trasách potrebné na vyhľadávanie najkratšej cesty.

Definícia 2 (Cesta)**.** *Cesta v grafe $G = (V,E)$ je postupnosť $P = (v_0,e_0,v_1, ...,e_{n-1},v_n)$, pre ktorú platí:*

- $\forall(e_i \in P) : e_i \in E$,
- $\forall(v_i \in P) : v_i \in V$,
- $\forall(e_i \in P) : e_i = \{v_i,v_{i+1}\}$ *(Matoušek a Nešetřil, 2009).*

Druhá komplikácia, s ktorou sa stretávame pri tvorbe grafu je, že nie každý bod tvoriaci cestu v mapových dátach musí byť aj vrcholom grafu (kap. 3.1). Pri vyhľadávaní sú podstatné iba tie body, ktoré sú buď na začiatku, alebo konci cesty grafu alebo sa nachádzajú na križovatke s inou cestou.

Orientovaný a neorientovaný graf

Táto práca sa zameriava na turistické mapy, a preto pri vyhľadávaní nebudú brané do úvahy pravidlá cestnej premávky, ako napríklad značka jednosmernej cesty alebo zákaz odbočenia doprava. To znamená, že po každej ceste je možné sa pohybovať oboma smermi, takže sa dá mapa reprezentovať ako neorientovaný graf.

Vyhľadávanie s výškovými dátami

Pri navigácií v turistických mapách je potrebné brať do úvahy aj výškové dáta, pretože na rozdiel od navigácie pre autá, výškový profil cesty dokáže výrazne ovplyvniť čas potrebný na jej prejdenie. Existuje viacero spôsobov ako upraviť vyhľadávanie najkratšej cesty tak, aby boli brané do úvahy aj výškové dáta. V tejto práci boli zvažované dva konkrétne prístupy:

- Na výpočet ohodnotenia hrany sa použije funkcia, ktorá berie do úvahy sklon danej cesty. Hlavnou výhodou takéhoto prístupu je, že je možné veľmi presne modelovať zmenu rýchlosti v závislosti od sklonu cesty. Napríklad, dokáže popísať situáciu, keď sme pri 30° stúpaní donútení zosadnúť z bicykla.

- Pre každú hranu sa uloží informácia o tom, koľko výškových metrov sa ide do kopca, a koľko dole z kopca. Následne stačí dovoliť užívateľovi nastaviť penaltu za každý výškový meter stúpania, a prípadne bonus za každý meter klesania. Na výpočet ohodnotenia hrany potom stačí k jej váhe bez výškových metrov pripočítať penalty a bonusy za klesanie a stúpanie. Hlavnou výhodou tohto prístupu je rýchlosť výpočtu ohodnotenia hrany. Jeho presnosť pri vhodnej voľbe konštánt, v priemernom prípade bude porovnateľná s prvým prístupom. Nevýhodou tohto prístupu samozrejme je, že nedokáže dobre modelovať rôzne rýchlosti pri rôznych stupňoch stúpania.

Definícia 3 (Ohodnotenie grafu)**.** *Ohodnotenie grafu je funkcia, ktorá každej hrane grafu priradí numerickú hodnotu. (Matoušek a Nešetřil, 2009)*

Táto aplikácia využíva druhý spôsob práce s výškovými dátami. Hlavný dôvodom je, že formát, ktorý aplikácia využíva na ukladanie informácií o mapových dátach (kap. 8.3), by pri prvom spôsobe vyžadoval uchovávanie výrazne väčšieho množstva informácií pre väčšinu hrán, ako pri použití druhého spôsobu.

Tým, že sa pri vyhľadávaní využívajú výškové dáta nastáva situácia, že aj napriek tomu, že sa v práci pracuje s neorientovaným grafom, jednotlivé hrany možu mať iné ohodnotenie jedným smerom ako tým druhým. Pretože jedným smerom sa ide hore a druhým zase dole z kopca.

0.2 Zobrazenie mapy

Na to, aby bolo možné v mape vyhľadávať, je potrebné zobraziť mapové dáta ako mapu, v ktorej sa dokáže človek orientovať. Sú dva základné spôsoby ako k zobrazovaniu mapy pristúpiť:

- Mapa sa vykreslí z mapových dát pomocou frameworku na základe užívateľských nastavení ako obrázok. Vďaka tomu je možné nastaviť čo, a ako, sa na mape zobrazí. Ďalšou výhodou tohto prístupu je, že ak sa použijú rovnaké mapové dáta pre vyhľadávanie a aj pre vykresľovanie, je zaručené, že sa nájdená cesta skutočne správne zobrazí na mape. Nevýhodou je, že vykresľovanie mapy je výkonnostne a časovo náročné.

- Namiesto vykresľovania mapy z mapových dát sa použijú už existujúce, niekym vykreslené dáta a tie aplikácia zobrazí užívateľovi pomocou tilingu (kap. 4.1). Tento prístup má výhodu v tom, že sa ušetrí netriviálne vykresľovanie mapy. Na druhú stranu nebude možné mapu prispôsobiť potrebám aplikácie. Ďalší problém, ktorý môže nastať je, že sa mapové dáta použité na vykreslenie mapy môžu líšiť od tých použitých na vyhľadávanie cesty.

0.3 Užívateľské rozhranie

Na vyhľadávanie najkratšej cesty, je potrebné získať vstup od užívateľa. Takýmto vstupom sú hlavne informácie odkiaľ a kam má viesť vyhľadaná cesta a aké parametre majú byť pri hľadaní najkratšej cesty použité. Parametre vyhľadávania v tejto práci sú rýchlosť pohybu a penalty za plusové a mínusové metre pre daný typ cesty.

To znamená, že je potrebné, aby aplikácia okrem samotného zobrazenia mapy taktiež povoľovala navigáciu po mape, priblíženie a oddialenie mapy, a aby bolo možné nájsť a zvoliť začiatočný a konečný bod cesty. Ďalej je potrebné vytvoriť vhodné prostredie, pomocou ktorého bude možné meniť nastavenia parametrov cesty. V neposlednom rade musí aplikácia ponúknuť možnosť vyhľadať najkratšiu cestu podľa navolených parametrov a po jej nájdení musí danú cestu vykresliť na zobrazenú mapu.

0.4 Výber platformy a programovacieho jazyka

Výber programovacieho jazyka, v ktorom bude aplikácia napísaná, je veľmi dôležitý, pretože sa od voľby bude odvíjať, aké knižnice a nástroje budú použité pri vývoji aplikácie a tiež to, pod ktorými platformami bude aplikácia spustiteľná.

C++

Jazyk C++ (Standard C++ Foundation, 2017) je objektovo orientovaná nadstavba jazyka C, ktorý bol prvý krát verejne predstavený v roku 1985[1]. V súčasnosti je najnovším štandardom jazyka štandard C++14[2] z roku 2014. Pre programy napísané v tomto štandarde existuje veľké množstvo prekladačov, vďaka čomu je možné tieto programy spustiť na všetkých populárnych platformách.

Hlavnou výhodou tohto programovacieho jazyka je, že obsahuje nástroje na prácu s dátami na veľmi nízkej úrovni a umožňuje veľmi efektívnu prácu s pamäťou počítača. Hlavnou nevýhodou je, že na vytváranie užívateľského prostredia

[1] http://www.stroustrup.com/bs_faq.html#invention
[2] https://isocpp.org/wiki/faq/cpp14-language

je potrebné buď komunikovať priamo s API[3] konkrétneho operačného systému alebo použiť externú knižnicu.

C#

Jazyk C# (Microsoft, 2017a) je objektovo orientovaný jazyk vyvinutý spoločnosťou Microsoft, ktorý vychádza z jazykov C++ a Java a beží pod .NET frameworkom (Microsoft, 2017b). Začiatkom roku 2017 bola vydaná verzia 7.0[4] jazyka a verzia 4.6.2[5] .NET frameworku.

Hlavnými dôvodmi voľby jazyka C# na implementáciu aplikácie je jednoduchá tvorba užívateľského prostredia, iba s použitím natívnych knižníc .NET frameworku a existencia knižnice, pomocou ktorej sa dá jednoducho spojiť toto užívateľské prostredie so zobrazovaním mapových dát (kap. 4.3).

Z dôvodu využitia knižníc, ktoré potrebujú .NET framework verzie 4.0 a vyššej, táto práca nie je prispôsobená na využitie na iných ako Windowsových[6] platformách. Na spustenie pod inými platformami je možné použiť Mono[7], ale nie je zaručené, že aplikácia bude správne pracovať.

Použitie .NET frameworku so sebou nesie nevýhodu, že veľkosť objektov je limitovaná na 2GB, čo pri mapách väčších rozmerov môže spôsobiť problémy.

0.5 Popis kapitol

V prvej kapitole sú predstavené existujúce aplikácie, ktoré sú zameraním podobné problému riešenému v tejto práci. V druhej kapitole sú predstavené zdroje mapových a výškových dát. Konkrétne formáty súborov týchto dát sú popísané v tretej kapitole. Štvrtá kapitola sa zaoberá vykresľovaním mapy a niektorými frameworkami, ktoré sú na to určené. V piatej kapitole sú popísané rôzne algoritmické prístupy k vyhľadávaniu najkratšej cesty v grafe. Dátové štruktúry, pomocou ktorých je možné ukladať informácie o grafe popisuje šiesta kapitola. Je v nej taktiež popísaná dátová štruktúra použitá v algoritme na vyhľadávanie najkratšej cesty. Posledné dve kapitoly tvorí užívateľská a programátorská dokumentácia.

[3] z anglického Application Programming Interface
[4] https://blogs.msdn.microsoft.com/dotnet/2016/08/24/whats-new-in-csharp-7-0/
[5] https://blogs.msdn.microsoft.com/dotnet/2016/08/02/announcing-net-framework-4-6-2/
[6] https://www.microsoft.com/en-us/windows
[7] http://www.mono-project.com/

1. Existujúce aplikácie

V nasledujúcej kapitole sa budeme venovať analýze aplikácií hľadajúcich najkratšiu cestu v skutočných mapách, ktoré sú v súčasnosti dostupné pre užívateľov. Existuje ich viacero a líšia sa v rôznych parametroch, napríklad zdrojmi mapových dát, štýlom vykresľovania mapy a v neposlednom rade parametrami vyhľadávania najkratšej cesty a možnosťami ich prispôsobenia.

1.1 Google Maps

Google Maps[1] je jednou z najznámejších a najpoužívanejších mapových služieb na svete. Ide o webovú aplikáciu, ktorá má back-end napísaný v C++ a vo front-ende využíva JavaScript (International, 2016), XML (W3C, 2017) a Ajax[2].

Google Maps boli až do roku 2004 vyvíjané ako desktopová aplikácia vo Where 2 Technologies. Následne firmu odkúpila firma Google[3] a pretvorila ich na webovú aplikáciu. Google Maps boli spustené vo februári roku 2005 a v roku 2008 bola spustená mobilná verzia tejto služby (Google, 2013).

Samotná aplikácia povoľuje užívateľovi si zvoliť buď klasické zobrazenie mapy, alebo zobrazenie vo forme satelitných snímok. Ďalej povoľuje vyhľadávanie miest, ako napríklad reštaurácie alebo obchody v blízkosti používateľa, ale aj konkrétnych adries a lokácií, ktoré následne na mape zobrazí.

Google Maps taktiež povoľujú vyhľadávanie najkratšej cesty, a to pre profily auto, hromadná doprava, chodec, bicykel a lietadlo. Navyše pri vyhľadávaní najkratšej cesty sú užívateľovi ponúknuté aj druhá a tretia najkratšia cesta (obr. 1.1).

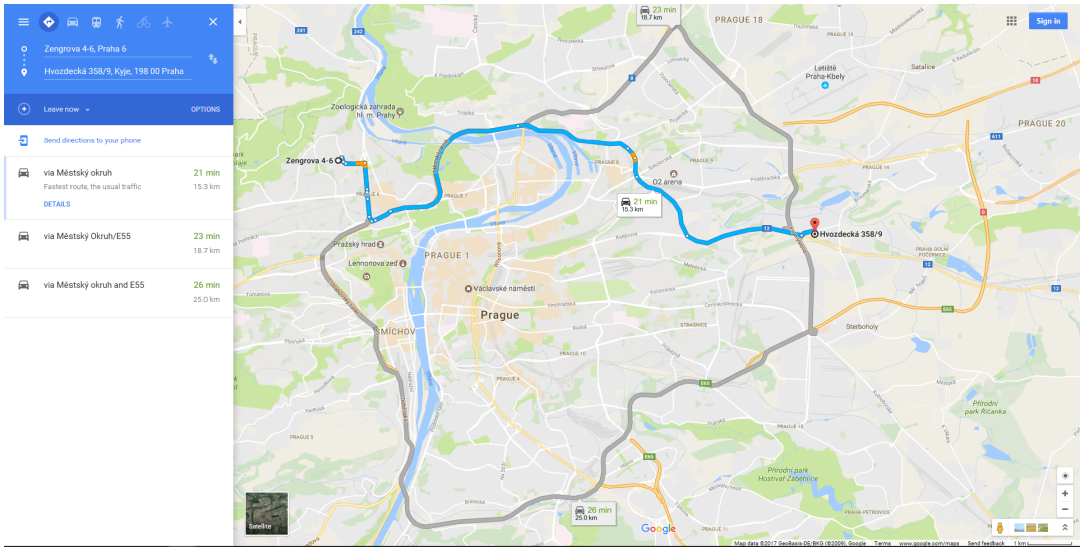
1.2 GraphHopper

GraphHopper[4] je open-source knižnica určená na navigáciu v mapách napísaná v jazyku Java (Oracle, 2017) a licencovaná pod Apache License 2.0 (Foundation, 2017). Implicitne využíva mapové podklady z projektu OpenStreetMaps (kap. 2.1) a výškové dáta z misie SRTM (kap. 2.2), ale podporuje aj iné zdroje dát.

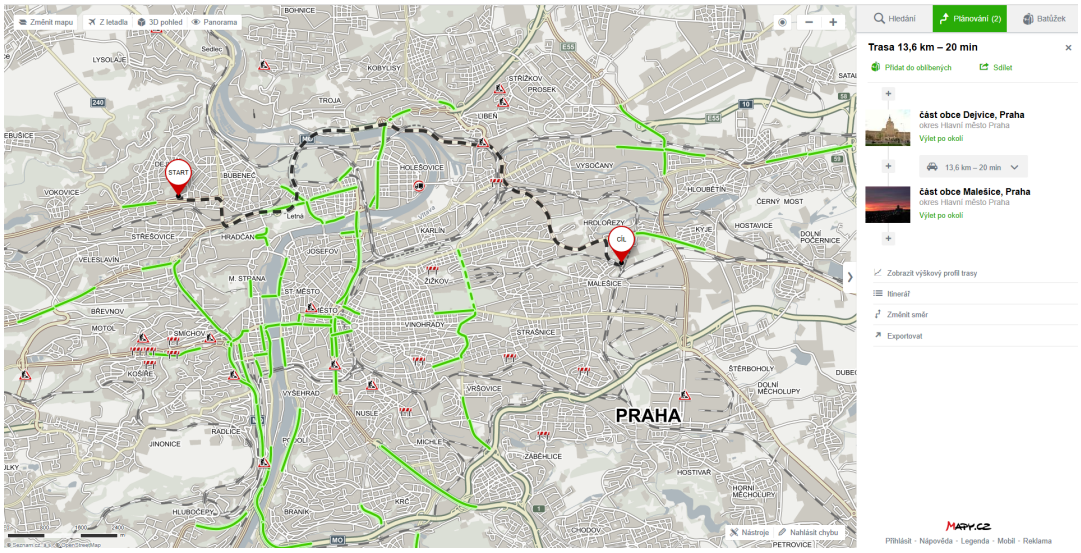
Na vyhľadávanie sa štandartne využíva algoritmus Contraction Hierarchies (kap. 5.3), ale knižnica sa dá nastaviť tak, aby využívala aj iné algoritmy, ako napríklad Dijkstrov (kap. 5.1) alebo A* (kap. 5.4).

Od februára 2015 využíva projekt OpenStreetMaps na vyhľadávanie najkratšej cesty pre bicykle a peších práve túto knižnicu.

[1] https://www.google.com/maps
[2] https://www.xul.fr/en-xml-ajax.html
[3] https://www.google.cz/intl/en/about/
[4] https://www.graphhopper.com/



Obr. 1.1: Ukážka vyhľadávania v google maps.



Obr. 1.2: Vyhľadávanie v aplikácií mapy.cz s použitím dopravného štýlu.

1.3 Mapy.cz

Mapy.cz[5] je webová mapová aplikácia vytvorená spoločnosťou Seznam[6]. Zameriava sa prevažne na mapy pre Českú a Slovenskú republiku, ale obsahuje mapové podklady pre celý svet. Dáta pre Česko a Slovensko čerpá z vlastných kartografických dát, zatiaľ čo pre ostatné lokality využíva mapové podklady z projektu OpenStreetMaps.

Aplikácia ponúka veľké množstvo štýlov zobrazenia mapy, vďaka ktorým môže užívateľ využívať to vykreslenie, ktoré sa zrovna najviac hodí jeho aktuálnym preferenciám. Medzi zaujímavé štýly vykreslenia patrí napríklad štýl letecká mapa, ktorý zobrazí mapu ako fotky z lietadla, alebo haptická mapa pre nevidiacich, ktorá sa dá vytlačiť v špecializovaných centrách, avšak tento typ mapy je do-

[5]https://mapy.cz/
[6]https://onas.seznam.cz/cz/o-firme/vize-firmy/

stupný iba pre územie Českej a Slovenskej republiky.

Okrem samotného zobrazovania mapy aplikácia umožňuje vyhľadávanie rôznych služieb, ako napríklad reštaurácií alebo zoologických záhrad, a tiež povoľuje vyhľadávanie najkratšej cesty podľa jedného zo šiestich rôznych profilov, ktorými sú auto, autobus, cyklista, turista, bežkár a kanoista.

2. Zdroje dát

Mapové dáta sú najdôležitejšou časťou práce, nakoľko od nich závisí niekoľko vlastností výslednej aplikácie. Kvalita mapových dát priamo ovplyvňuje kvalitu vykreslenia mapy a aj vyhľadávania cesty. Formát, v ktorom mapové dáta získame, určuje, aké úpravy sú potrebné na získanie formátu vhodného na prácu s dátami v aplikácii. Licencia, pod ktorou sú dáta distribuované, je rovnako veľmi dôležitá, keďže pre potreby tejto práce je potrebné aby išlo o voľne dostupné dáta.

Rôzne aplikácie majú rôzne požiadavky na mapové podklady. Aplikácia určená na vyhľadávanie trás pre autá bude požadovať najmä kvalitné dáta o cestách pre automobily. Na druhú stranu, nástroj, ktorý slúži k skúmaniu prírody, môže potrebovať kvalitné satelitné snímky a kvalita automobilových ciest nebude pre tento nástroj až tak dôležitá.

Pre ciele tejto práce sú potrebné mapové podklady s kvalitne zmapovanými turistickými a cyklistickými cestami, ktoré je možné vhodne skombinovať s výškovými dátami.

2.1 OpenStreetMaps

Pre potreby tejto práce boli zvolené dáta z OpenStreetMaps[1]. Ide o dobrovoľníkmi tvorený projekt, zameraný na vytvorenie kvalitných a voľne dostupných mapových dát. Samotné mapy sú z časti tvorené dátami poskytnutými rôznymi inštitúciami, ale prevažne sú vypracované dobrovoľníkmi prispievajúcimi do projektu.

V posledných pár rokoch sa tieto mapové dáta tešia čoraz väčšej popularite, čo má za následok aj výrazné zlepšovanie kvality dát.

Samozrejme, to že sa jedná o dobrovoľníkmi tvorenom projekte sebou nesie aj nevýhody. Keďže veľa užívateľov nemá skúsenosti s označovaním ciest a samotné označovanie má veľké množstvo rôznych pravidiel, vznikajú v mapových dátach chyby a nekonzistentnosti. Taktiež je celý projekt vytvorený s predpokladom, že každý prispievateľ chce mapové dáta len zlepšovať a nemá obmedzenia v tom, aké zmeny je možné robiť. Toto umožňuje používateľom naschvál robiť nepravdivé zmeny.

2.2 Zdroje výškových dát

Ďalší typ dát tvoriacich podstatnú časť tejto aplikácie sú výškové dáta. Hlavným dôvodom je, že pri vyhľadávaní cesty pre turistov výškový profil cesty výrazne ovplyvňuje čas, ktorý bude trvať cestu zdolať.

Od začiatku 21. storočia bolo podniknutých viacero misií s úlohou zmapovať výškové informácie o planéte Zem.

[1] http://www.openstreetmap.org/about

ASTER

ASTER[2] je Japonský senzor, ktorý Americká agentúra pre letectvo a vesmír[3] v roku 1999 vypustila do zemského orbitu na satelite Terra. Mimo iného sa tento senzor využíva na vytvorenie detailných výškových dát.

V roku 2009 bol výškový model zeme sprístupnený verejnosti a obsahoval pokrytie 99% zemského povrchu. Aj keď boli dáta z tohto projektu získané vo vyššom rozlíšení ako pri misii SRTM, existujú názory, že skutočné rozlíšenie kvalitnejšie nie je, a navyše dáta obsahovali výrazné artefakty (Hirt, Filmer a Featherstone, 2010). Tieto nedostatky čiastočne vyriešila druhá verzia tohto modelu vydaná v roku 2011, avšak artefakty sa v dátach stále vyskytujú (de Ferranti).

AW3D30

AW3D30 je dataset vytvorený a zadarmo poskytovaný Japonskou vesmírnou agentúrou[4]. Ide o projekt s najvyššou presnosťou dát zo všetkých tu zmienených projektov. Dáta tohto projektu boli získané z obrázkov získaných Satelitom DAICHI.

SRTM

Shuttle Radar Topography Mission (NASA, 2017), je misia, ktorá bola v roku 2000 podniknutá úradom NASA a ktorej úlohou bolo získať výškové dáta o skoro celom svete. Dáta sú dostupné v troch verziách:

- Verzia 1.0. Jedná sa o dáta, ktoré sú skoro zhodné tým získaným priamo z misie.
- Verzia 2.1. V tejto verzii boli k dátam z verzie 1.0 pridané vodné plochy a navyše boli následne vyčistené od niektorých chýb.
- Verzia 3.0. Dáta, ktoré vznikli z verzie 2.1 doplnením chýbajúcich výškových dát z iných zdrojov.

Hlavný dôvod, prečo boli dáta z tohto projektu zvolené v tejto aplikácii, je, že sú využívané priamo projektom OSM, a preto, existujú viaceré nástroje na spracovanie dát v tomto formáte.

[2] z anglického Advanced Spaceborne Thermal Emission and Reflection Radiometer
[3] Z anglického National Aeronautics and Space Administration (skrátene NASA)
[4] Skrátene JAXA

3. Formáty OSM dát

V tejto kapitole sa budeme venovať rôznym formátom, do ktorých sa dajú dáta z projektu OpenStreetMaps upraviť a nástrojom na to určeným. Použitie jednotlivých programov používaných v tejto práci je popísané v užívateľskej dokumentácii (kapitola 7).

3.1 OSM XML

OSM XML[1] je jeden z dvoch základných formátov, v ktorých sa dajú dáta z OpenStreetMaps získať. Súbor typu OSM XML má príponu .osm, a je to XML súbor s pevne definovanou štruktúrou. Celý dokument je v podstate iba zoznam základných elementov, ktorými sú Vrchol, Cesta a Relácia.

Medzi výhody tohoto formátu patrí ľahká čitateľnosť človekom, jednoduché upravovanie, nezávislosť na platforme a fakt, že povoľuje ľahké spájanie viacerých súborov dohromady.

Hlavné nevýhody tohto formátu sú jeho priestorová náročnosť a pomalé vyhľadávanie údajov. Prvý problém vzniká priamo z toho, že je použitý formát XML a ten je obšírny. Tento problém je možné čiastočne eliminovať kompresiou dát. Napríklad veľkosť súboru s dátami o Českej republike je po kompresii do formátu .gz pomocou programu GNU Gzip[2] viac než 10 násobne menšia.

Vrchol

Element vrchol sa využíva na definovanie bodu v priestore. Každá inštancia typu vrchol musí mať ID, zemepisnú šírku a dĺžku. Taktiež môže mať príznaky popisujúce o aký objekt sa jedná.

Vrcholy sú určené na zaznačenie konkrétnych objektov, ako napríklad stromu, alebo vrcholu kopca, a aj na opis tvaru cesty. Posledný spomínaný typ vrcholov zväčša nemá žiadne príznaky.

Cesta

Cestu tvorí usporiadaný zoznam 2 až 2000 vrcholov, ktorý definuje tvar cesty. Tvar cesty sa dá získať pospájaním jednotlivých vrcholov zoznamu, v poradí od prvého do posledného. To umožnuje pomocou cesty, zaznačiť aj veľmi komplikované geometrické útvary.

Nevýhodou takéhoto značenia je, že komplikuje vyhľadávanie najkratšej cesty, keďže každá zákruta na trase rozdelí cestu na viacero celkov. Tento problém je možné takmer úplne eliminovať vhodným predspracovaním grafu (kap. 8.3).

Cesta môže začínať a končiť tým istým bodom. Takýmto spôsobom sa zaznačujú oblasti (napríklad lesy) ale aj čiary kruhového tvaru (napríklad kruhové objazdy).

Rovnako ako vrchol, aj cesta môže, ale nemusí, mať príznaky popisujúce objekt, ktorý cesta reprezentuje.

[1] http://wiki.openstreetmap.org/wiki/OSM_XML

[2] https://www.gnu.org/software/gzip

Relácia

Posledným z troch hlavných elementov OSM XML formátu je relácia. Používa sa na zdokumentovanie vzťahov medzi viacerými vrcholmi a/alebo cestami. Relácia môže predstavovať napríklad diaľnicu D1.

Príznaky

Príznaky slúžia na zdokumentovanie informácií o konkrétnom Vrchole, Ceste alebo Relácii. Napríklad pri cestách príznak popisuje, o aký typ cesty sa jedná. Cesta s príznakom highway a hodnotou motorway popisuje diaľnicu.

Všetky príznaky a ich vysvetlenia sú popísané na stránke OSM projektu[3].

3.2 OSM JSON

Formát OSM JSON vznikol prevedením z pôvodného XML formátu do JSON (International, 2013) syntaxe. Jediným podstatným rozdielom oproti OSM XML je menšia priestorová náročnosť.

3.3 Shapefile

Shapefile (ESRI, 1998) je dátový formát slúžiaci na ukladanie vektorových, priestorových a geografických informácií. Formát je schopný popísať vektorové útvary, ktorými sú body, čiary a mnohouholníky.

Samotný formát pozostáva z kolekcie súborov, ktoré majú rovnaký názov (okrem prípony) a sú uložené v rovnakom priečinku.

Hlavný problém Shapefile formátu je, že nedokáže ukladať topologické informácie.

3.4 PBF

Formát PBF[4] je určený ako alternatíva k OSM XML formátu. V porovnaní s OSM XML formátom je menší, a to dokonca aj po kompresii pôvodného OSM XML súboru. Taktiež má výrazne rýchlejšie čítanie a zapisovanie ako komprimovaný súbor, a rovnako ako pôvodný OSM XML formát, je možné súbory, vo PBF formáte ľahko rozširovať o nové dáta. Jeho nevýhodou je, že sa jedná o binárny formát, takže je náročné ho čítať voľným okom. Tento formát využíva napríklad projekt Osmosis (Osmosis, 2016).

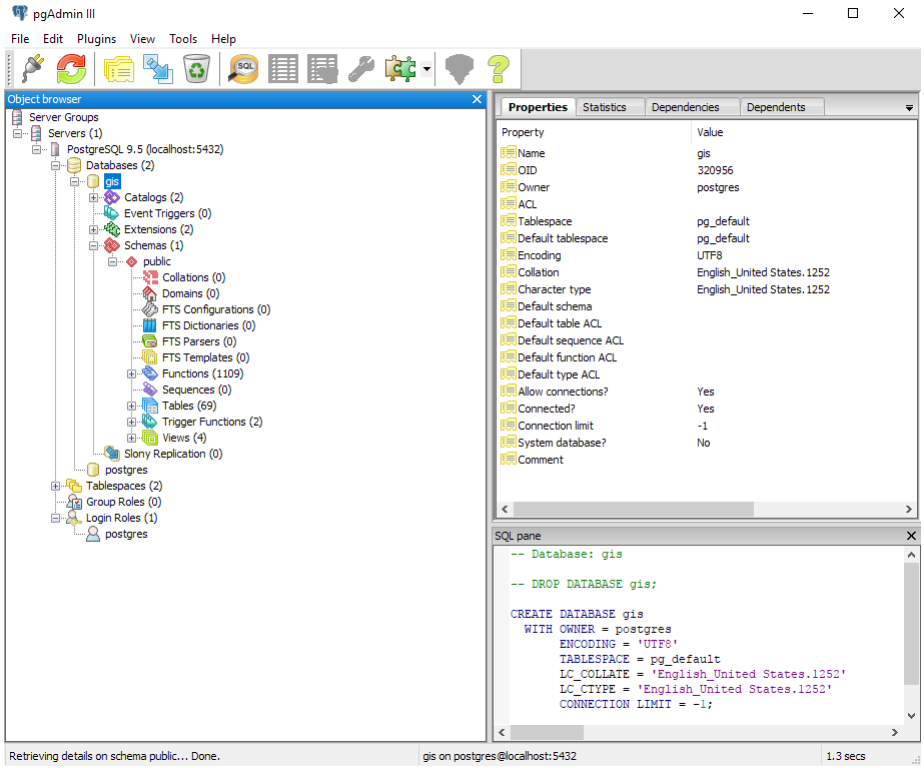
3.5 Priestorová Databáza

Veľmi častým spôsobom práce s OSM dátami je využitie databázy, ktorá dokáže efektívne pracovať s priestorovými typmi dát. Samotný projekt OSM využíva takto uložené dáta pre vykresľovanie máp[5].

[3] http://wiki.openstreetmap.org/wiki/Map_Features

[4] Z anglického Protocolbuffer Binary Format

[5] http://wiki.openstreetmap.org/wiki/Mapnik



Obr. 3.1: Ukážka prostredia aplikácie PGAdmin III.

PostgreSQL

PostgreSQL, pôvodne Postgres[6], je multiplatformový, open-source, objektovo-relačný databázový systém licencovaný pod PostgreSQL Licence (PostgreSQL, 2017).

Samotný PostgreSQL nedokáže veľmi dobre pracovať s priestorovými dátami. Tento problém rieši rozšírenie PostGIS[7], ktoré do PostgreSQL pridáva veľké množstvo funkcií[8] pre prácu s týmto typom dát. Toto rozšírenie je licencované pod GNU General Public Licence version 2 (Foundation, 1991).

Na réžiu PostgreSQL databázy sa dá použiť buď nástroj psql, ktorý beží na príkazovom riadku, alebo voľne dostupný software pgAdmin[9] distribuovaný pod The PostgreSQL Licence, rovnako ako samotný PostgreSQL.

Medzi výhody PostgreSQL patrí, že tento databázový systém využíva aj samotný projekt OSM a preto existuje viacero nástrojov, ktoré sú schopné importovať dáta vo formáte OSM XML do PostgreSQL databázy.

Osm2pgsql

Osm2pgsql (osm2pgsql, 2017) je program bežiaci na príkazovom riadku, slúžiaci k importu dát z formátu OSM XML do PostgreSQL databáze s PostGIS rozšírením.

[6]https://www.postgresql.org/about/history/
[7]https://postgis.net/
[8]https://postgis.net/features/
[9]https://www.pgadmin.org/index.php

Má dva základné módy - normálny a slim. Slim mód používa pomocné tabuľky, ktoré si ukladá v databáze na disku. Normálny mód využíva pamäť RAM, čo znamená, že pri importoch rozsiahlych máp je potrebné spúšťať 64-bitovú verziu Osm2pgsql a mať dostatočné množstvo RAM pamäte v počítači.

Tento program je prispôsobený pre platformy Ubuntu[10] a distribúcií založených na Debiane[11]. Na spustenie tohto programu na platforme Windows je najjednoduchšie použiť balíček Cygwin[12].

3.6 Výškové dáta

Výškové dáta z misie SRTM sú distribuované vo vlastnom type formátu. Pre prácu s týmito dátami je nutné porozumieť tomuto formátu, tomu ako sa z neho dajú získať informácie o výške daného bodu a tomu ako sa z týchto dát dajú vytvoriť kontúry, ktoré bude možné vykresliť na mape.

Formát HGT

Zmapovaný povrch zeme bol rozdelený do mriežky, ktorá bola následne rozdelená do konkrétnych súborov. Tieto súbory sú dodávané vo formáte HGT. Ide o veľmi jednoduchý a kompaktný formát. Celý súbor je vlastne postupnosť po sebe idúcich 16 bitových čísel, ktoré udávajú výšku konkrétneho políčka mriežky.

Srtm2Osm

Srtm2Osm[13] je nástroj bežiaci na príkazovom riadku, určený na vytváranie kontúr vo formáte OSM XML, z dát získaných z projektu SRTM.

Tento nástroj beží pod platformami Windows a s nástrojom Mono dokáže bežať aj pod Linuxovými[14] platformami. Nástroj má tiež veľké množstvo rôznych nastavení, ktorými sa dá ovplyvniť formát výstupných dát. Samotné dáta z projektu SRTM si nástroj sťahuje sám z NASA FTP serveru.

Phyghtmap

Phygtmap[15] je vylepšená verzia Srtm2Osm. Ide o nástroj napísaný v Pythone, ktorý rovnako ako Srtm2Osm stiahne SRTM dáta z NASA serveru, a vytvorí z nich kontúry vo formáte OSM XML. V porovnaní s Srtm2Osm je tento nástroj rýchlejší a kvalitne udržovaný, čo je aj dôvod prečo bol v tejto práci využitý na vygenerovanie kontúr.

[10] https://www.ubuntu.com/
[11] https://www.debian.org/
[12] https://www.cygwin.com/
[13] http://wiki.openstreetmap.org/wiki/Srtm2Osm
[14] https://www.linux.org/
[15] http://katze.tfiu.de/projects/phyghtmap/

4. Vykresľovanie mapy

V tejto kapitole budú predstavené rôzne frameworky, pomocou ktorých sa dá vykresliť a zobraziť mapa.

4.1 Tiling

Vykresľovanie mapy je výkonovo a časovo náročné a často sa stáva, že užívateľ potrebuje pri navigácii viac krát vykresliť tú istú oblasť mapy. Z týchto dôvodov mapové aplikácie využívajú na zobrazovanie predgenerované obrázky mapy, ktoré sa nazývajú dlaždice[1]. Štandardne každá dlaždica je 256×256 obrázok vo formáte PNG, ktorý má priradené špeciálne súradnice X,Y[2].

Všetky dlaždice bývajú usporiadané v adresárovej štruktúre, v ktorej každá úroveň priblíženia[3] a každý stĺpec má vlastný priečinok a každá dlaždica v danom stĺpci je súbor v danom priečinku. Dlaždica so súradnicami (X,Y) pri priblížení Z má v adresárovej štruktúre cestu: /Z/X/Y.png.

4.2 Mapnik

Mapnik[4] je open-sourcový nástroj, napísaný v jazyku C++, bežiaci na príkazovom riadku určený ku generovaniu máp. Dokáže využívať rôzne formáty vstupných dát, napríklad Shapefile, OSM XML alebo PostgreSQL. To, ako sa mapa vykreslí, sa dá nastavovať podľa štýlov, väčšinou v XML formáte špecifickom pre Mapnik.

Projekt OSM využíva Mapnik na generovanie dlaždíc, ktoré potom zobrazuje pomocou JavaScriptu vo webovej aplikácií.

4.3 SharpMap

SharpMap (SharpMap, 2017) je knižnica napísaná v jazyku C#, určená na vykresľovanie a zobrazovanie mapových dát vo webových a desktopových aplikáciách. Je licencovaná pod GNU Lesser General Public Licence (Foundation, 2007) a poskytuje podporu pre rôzne zdroje mapových dát, ako napríklad PostgreSQL a Oracle[5] databázové systémy.

Knižnica dokáže sama vykresliť mapu z mapových podkladov, ale má aj podporu pre zobrazovanie už existujúceho tilingu.

Samotné vykresľovanie mapy funguje po vrstvách. Knižnici sa predá zoznam vrstiev a tá ich v poradí vykreslí do mapového okna. Pri tomto procese záleží na poradí vrstiev. Ak sa napríklad prvé dajú vykresliť cesty a potom lesy, stratí sa informácia o všetkých cestách, ktoré ležia v lese.

[1] Z anglického Tiles.
[2] http://wiki.openstreetmap.org/wiki/Slippy_map_tilenames#X_and_Y
[3] http://wiki.openstreetmap.org/wiki/Zoom_levels
[4] http://mapnik.org/index.html
[5] https://www.oracle.com/database/index.html

Hlavný dôvod, prečo je v tejto práci využívaný tento framework je, že povoľuje aj vykresľovanie vlastnej mapy, ale aj zobrazovanie už existujúceho tilingu, a navyše sa dá jednoducho použiť vo windows forms aplikácií.

5. Vyhľadávanie v mapách

Hľadanie najkratšej cesty v grafe je veľmi populárnym grafovým problémom. Existuje veľké množstvo variácií tejto úlohy (Turner, 2011). Táto práca sa zameriava na problém hľadania najkratšej cesty medzi dvoma bodmi v ohodnotenom, neorientovanom grafe.

Pretože sa v práci využíva reprezentácia výškových dát pomocou plusových a mínusových metrov, výškové dáta ovplyvnia iba ohodnotenie jednotlivých hrán a nie je nutné modifikovať algoritmus vyhľadávania najkratšej cesty.

Pre potreby tejto práce sú potrebné varianty algoritmu a dátových štruktúr, ktoré dokážu pracovať s veľkým množstvom dát, povoľujú rýchlu zmenu parametrov vyhľadávania a sú dostatočne rýchle, aby vyhľadávanie bežalo v reálnom čase.

5.1 Dijkstrov algoritmus

Asi najznámejším algoritmom na riešenie problému najkratšej cesty z jedného bodu do druhého je Dijkstrov algoritmus (Dijkstra, 1959). Tento algoritmus funguje na všetkých grafoch s kladným ohodnotením hrán.

Algoritmus pri inicializácii vloží všetky vrcholy do množiny otvorených vrcholov, nastaví im predchádzajú vrchol na nedefinovaný a vzdialenosť im všetkým okrem začiatočného nastaví na nekonečno. Začiatočnému vrcholu je nastavená vzdialenosť na nulu. Vzdialenosť vyjadruje dĺžku najkratšej nájdenej cesty zo začiatku do daného vrcholu a informácia o predchádzajúcom vrchole sa využíva k získaniu najkratšej cesty.

Po inicializácií algoritmus vyextrahuje vrchol s minimálnou vzdialenosťou z otvorenej množiny a spracuje ho. Toto opakuje až pokiaľ nenarazí na cieľ. Pri spracovaní vrcholu v algoritmus vezme všetkých susedov z otvorenej množiny tohto vrcholu, t.j. vrcholy, ktoré sú s v spojené hranou a skontroluje, či cesta, ktorá by viedla cez vrchol v do daného susedného vrcholu nie je kratšia ako tá, ktorá bola nájdená doteraz. V prípade, že cesta cez vrchol v je kratšia, nastaví vzdialenosť vrcholu na veľkosť tejto cesty a ako predchodcu mu dá vrchol v. Táto procedúra sa niekedy nazýva pojmom relaxácia.

Po dobehnutí algoritmu sa najkratšia cesta zo začiatku do cieľa, ak existuje, dá získať vo forme zoznamu nasledovne. Cieľ sa nastaví ako vrchol s, a následne sa spracováva s až pokiaľ s nie je nedefinovaný. Pri spracovaní s je s pridaný na začiatok zoznamu a do s sa dosadí predchádzajúci vrchol. Za s sa dosadí nedefinovaný vrchol až po spracovaní začiatočného vrcholu. To či cesta do cieľa existuje sa dá skontrolovať tak, že sa skontroluje vzdialenosť. V prípade, že vzdialenosť nie je nastavená na nekonečno algoritmus našiel najkratšiu cestu.

Nižšie je ukázaný pseudokód tohto algoritmu pre vyhľadanie najkratšej cesty medzi dvoma vrcholmi (source a target), ktorý využíva prioritnú frontu (kap. 6.2) na implementáciu množiny otvorených vrcholov. V prípade, že je úlohou nájsť najkratšiu cestu zo začiatku do všetkých vrcholov, stačí odstrániť riadky 11 a 12, a teda algoritmus bude bežať až do vtedy, pokiaľ nebude otvorená množina prázdna. Na zistenie najkratšej cesty zo začiatku do konkrétneho vrcholu stačí použiť funkciu Extract_path pre ten konkrétny vrchol.

```
1.  List<Vertex> Dijsktra(Graph graph, Vertex source, target)
2.  Priority_queue<Vertex> open;
3.  HashSet<Vertex> closed;
3.  foreach(vertex in graph.Get_vertices())
4.     vertex.Distance = infinity;
5.     vertex.Predecessor = Vertex.Undefined;
6.     open.Add(vertex);
7.  open.Decrease_distance(source, 0);
8.
9.  while(not(open.Empty))
10.    current = open.Extract_min();
11.    if(current == target)
12.       return Extract_path(graph, target);
13.    closed.Add(current);
14.    foreach(vertex in graph.Get_neighbours(current))
15.       if(closed.Contains(vertex))
16.          continue;
17.       dist = current.Distance +
18.              graph.Get_edge(current, vertex).Length;
19.       if(dist < vertex.Distance)
20.          open.Decrease_distance(vertex, dist);
21.          vertex.Predecessor = current;
23. return null;
24.
25. List<Vertex> Extract_path(Graph graph, target)
26. if(target.distance == infinity)
27.    return null;
28. List<Vertex> path;
29. current = target;
30. do
31.   path.Add(current);
32.   current = current.Predecessor;
33. while(not(current == Vertex.Undefined))
34. return path;
```

Ako je ukázané v pseudokóde algoritmu, operácie, ktoré sú vykonávané nad množinou otvorených vrcholov, sú Add, Extract_min a Decrease_distance. Operácia Add pridá prvok do množiny, Extract_min odstráni prvok, ktorý má minimálnu vzdialenosť (distance) z množiny a operácia Decrease_distance zníži vzdialenosť vrcholu, čím sa môže zmeniť štruktúra otvorenej množiny.

Asymptotická zložitosť algoritmu priamo záleží na výbere dátovej štruktúry, pomocou ktorej sa implementuje množina otvorených vrcholov.

Hlavnou nevýhodou tohoto algoritmu je, že nedokáže využiť vlastnosti grafu na zrýchlenie vyhľadávania. Jeho výhodami sú jednoduchá implementácia, jeho veľmi dobrá asymptotická zložitosť a to, že nepotrebuje predspracovanie grafu.

5.2 Thorup

V roku 1999 predstavil Dánsky matematik Thorup algoritmus založený na Dijkstrovom algoritme, určený na vyhľadávanie najkratšej cesty v neorientovaných grafoch s celočíselným ohodnotením hrán v lineárnom čase a za použitia lineárneho priestoru (Thorup, 1999). Používa priehradkovú dátovú štruktúru, vďaka ktorej je schopný vyhnúť sa používaniu porovnávacieho triediaceho algoritmu.

Pre túto prácu tento algoritmus nie je vhodný, nakoľko požiadavka na celočíselné váhy hrán by výrazne skomplikovala hľadanie najkratšej cesty.

5.3 Contraction Hierachies

Algoritmus Contraction Hierachies využíva predspracovanie grafu na výrazné zrýchlenie samotného vyhľadávania. Hlavná výhoda tohto algoritmu je v tom, že umožňuje extrémne rýchle vyhľadávanie najkratšej cesty aj bez použitia heuristiky. Nevýhoda tohto algoritmu je, že potrebuje predspracovať graf, čo je vysoko časovo a pamäťovo nákladné. Navyše je predspracovanie znova potrebné na zmenu profilu vyhľadávania.

Poslednú spomínanú nevýhodu čiastočne rieši algoritmus Customizable Contraction Hierachies (Dibbelt, Strasser a Wagner, 2015), ktorý má dve fázy predspracovania. Pričom váhy hrán sú brané do úvahy až v druhej fáze, ktorá je relatívne rýchla a pri zmene profilu vyhľadávania stačí znova spustiť túto fázu.

5.4 A*

Algoritmus A* (Hart, Nilsson a Raphael, 1968) vznikol ako vylepšenie Dijkstrovho algoritmu pridaním heuristiky na voľbu vhodného vrcholu z množiny otvorených vrcholov. Výber vhodnej heuristiky je veľmi dôležitý. Na to, aby cesta nájdená A* algoritmom bola optimálna, zvolená heuristika musí byť prípustná[1] a monotónna.

Algoritmus A* sa od Dijkstrovho líši tým, že okrem takzvaného G-Skóre (dĺžka najkratšej cesty zo začiatku do daného vrcholu) si uchováva aj takzvané F-Skóre, čo je dĺžka najkratšej cesty, ktorá ide zo začiatočného do konečného bodu a prechádza cez daný bod. Toto skóre je vlastne súčet G-Skóre vrcholu a hodnoty heuristickej funkcie v danom vrchole. Pri voľbe vrcholu z otvorenej množiny sa berie ten s najmenším F-Skóre.

Definícia 4 (Prípustná heuristika)**.** *Nech $h(n)$ je hodnota heuristického odhadu funkcie vo vrchole n, a nech $h^*(n)$ je skutočná hodnota funkcie vo vrchole n. Povieme, že heuristika h je prípustná, ak $\forall(n) : h(n) \leq h^*(n)$. (Russell a Norvig, 2002)*

Definícia 5 (Monotónna heuristika)**.** *Nech $h(n)$ je hodnota heuristického odhadu funkcie vo vrchole n, nech x,y sú susedné vrcholy, a nech $c(x,y)$ je váha hrany medzi x,y. Povieme, že heuristika h je monotónna, ak $\forall(x,y) : h(x) \leq c(x,y) + h(y)$. (Russell a Norvig, 2002)*

[1] Z anglického Admissable heuristic

V prípade, že zvolená heuristika nemá požadované vlastnosti, algoritmus nemusí nájsť optimálnu cestu. To, do akej miery sa môže výsledok líšiť od optima, závisí na vlastnostiach zvolenej heuristiky, čo využívajú niektoré aproximačné algoritmy, ako napríklad AplhaA* (Reese, 1999) aby veľmi rýchlo získali cestu ktorá je najviac $1 + \varepsilon$ násobne dlhšia ako cesta optimálna.

V tejto práci sa na vyhľadávanie najkratšej cesty využíva A*, pretože nevyžaduje predspracovanie, funguje ako na orientovaných, tak aj na neorientovaných grafoch a na reálnych mapových dátach by mal rýchlosťou prekonať Dijsktrov algoritmus.

Ako heuristika je v projekte použitá vzdialenosť bodu od cieľa cesty, vydelená maximálnou rýchlosťou, ktorou sa dá pohybovať. Toto delenie je potrebné, pretože bez neho by nemuseli platiť požiadavky, ktoré sú kladené na heuristickú funkciu, a nájdená cesta by nemusela byť optimálna.

Pretože sa v projekte pri vyhľadávaní využívajú aj výškové dáta, môžu sa porušiť vlastnosti použitej heuristiky. Bolo by možné vytvoriť heuristickú funkciu, ktorá by splnila požadované vlastnosti aj pri použití výškových dát. Táto heuristika by ale bola výrazne „slabšia" ako tá, ktorá bola použitá v práci, čo by malo za následok spomalenie rýchlosti vyhľadávania. Podľa názoru autora, pri typickom nastavení parametrov cesty možné porušenie vlastností použitej heuristiky nespôsobí významnú odchýlku od optimálnej cesty.

6. Dátové štruktúry

Mapové dáta, v pôvodnom formáte OSM XML, nie sú prispôsobené na vyhľadávanie najkratšej cesty. Obsahujú veľké množstvo nepotrebných informácií a navyše formát XML má veľké priestorové požiadavky. Preto je ich potrebné upraviť do formátu, ktorý je vhodnejší na účely tejto aplikácie. Okrem dátovej štruktúry určenej na reprezentáciu grafu, je podstatná aj voľba vhodných dátových štruktúr na implementáciu algoritmu na vyhľadávanie najkratšej cesty.

6.1 Reprezentácia grafu

Pri výbere vhodnej dátovej štruktúry na reprezentáciu grafu na vyhľadávanie najkratšej cesty boli v tejto práci brané do úvahy dve hlavné vlastnosti - priestorová náročnosť a čas, ktorý je potrebný na nájdenie susedov daného vrcholu.

Zoznam susedov

Zoznam susedov je dátová štruktúra, ktorá sa často používa pri implementácii Dijkstrovho algoritmu. Je to kolekcia vrcholov, v ktorej každý vrchol má odkaz na zoznam k nemu susedných vrcholov.

Tento spôsob reprezentácie umožňuje veľmi rýchle získanie všetkých susedov konkrétneho vrcholu. Hlavnou nevýhodou tejto dátovej štruktúry je, že v prípade neorientovaného grafu je informácia o tom, že dva vrcholy spolu susedia, uložená dvakrát.

Vylepšený zoznam susedov

Vyššie spomínanú nevýhodu zoznamu susedov rieši dátová štruktúra, ktorú využíva projekt GraphHopper.

Tento formát je tvorený zoznamom vrcholov, v ktorom každý vrchol má odkaz na práve jednu hranu, ktorej je súčasťou. Každá hrana nesie informáciu o tom, ktorými vrcholmi je tvorená. Označme a, b vrcholy danej hrany, potom táto hrana nesie tiež informáciu o ďalšej hrane tvorenej vrcholom a a to isté pre vrchol b. Posledná hrana zoznamu ukazuje na prvú hranu zoznamu. To znamená, že všetky hrany, ktoré obsahujú daný vrchol, tvoria cyklický zoznam, v ktorom je podstatné, aby sa každá hrana nachádzala práve raz.

Pri správnej implementácii, je táto dátová štruktúra veľmi kompaktná a umožňuje rýchly prístup k informáciám o vrcholoch a aj o všetkých susedoch daného vrcholu. Z týchto dôvodov sa táto štruktúra využíva aj v tejto práci (kap. 8.3).

6.2 Prioritná fronta

Na implementáciu množiny otvorených vrcholov algoritmu A* je potrebné použiť dátovú štruktúru, ktorá dovoľuje operácie vkladanie prvku, získanie prvku s najnižším F-skóre a zmenu F-skóre už pridaného prvku. Tieto operácie sú štandardné pre abstraktný dátový typ prioritná fronta.

Častou dátovou štruktúrou, ktorou je implementovaná prioritná fronta, je halda. Halda je stromová dátová štruktúra pre ktorú platí, že každá dvojica rodič, syn má rovnaké usporiadanie v celej halde. V prípade minimovej haldy hodnota kľúča rodiča je vždy menšia alebo rovná hodnote kľúča syna.

V tejto práci sa pod pojmom halda bude myslieť dátová štruktúra, a nie oblasť pamäte určená na dynamickú alokáciu pamäte, ktorá sa rovnako nazýva halda.

Fibonacciho halda

Fibonacciho halda je dátová štruktúra, ktorú v roku 1987 predstavili Michael L. Fredman and Robert E. Tarjan (Fredman a Tarjan, 1987).

Táto dátová štruktúra má veľmi dobrú asymptotickú zložitosť jednotlivých operácii, avšak je náročná na efektívnu implementáciu a asymptotická zložitosť v sebe skrýva celkom podstatné konštanty, kvôli čomu Fibonacciho halda v reálnych implementáciách nedosahuje až také dobré výsledky ako iné dátové štruktúry (Saunders, 1999).

Binárna halda

Minimová binárna hlada je binárny stromu, ktorý má nasledujúce vlastnosti:

- Hodnota kľúča každého vrcholu je menšia ako hodnota kľúča každého vrcholu v jeho podstrome.
- Každá vrstva stromu okrem poslednej je plne zaplnená a v prípade, že posledná vrstva nie je plne zaplnená tak je táto vrstva čiastočne zľava súvisle zaplnená.

Veľkou výhodou tejto dátovej štruktúry je to, že sa dá implementovať pomocou poľa. Napríklad nasledovne, koreň stromu bude mať index 0, a deti vrcholu n budú na pozícii $2n+1$ a $2n+2$, potom rodiča vrcholu i nájdeme na pozícii $dlc(i-1)/2$, kde dlc je funkcia, ktorá vráti dolnú celú časť čísla.

V tejto práci bola zvolená táto dátová štruktúra preto, že sa dá jednoducho a efektívne implementovať a fakt, že mapové dáta tvoria poväčšine veľmi riedky graf na ktorom sa operácia zníž hodnotu prvku volá veľmi zriedka. Táto operácia je asymptoticky pomalšia v binárnej halde ako v iných podobných dátových štruktúrach. Ďalšia výhoda tejto štruktúry je, že z dôvodu implementácie pomocou poľa dokáže prekladač výrazne optimalizovať kód.

7. Užívateľská dokumentácia

Táto kapitola bližšie popisuje inštaláciu a užívateľské prostredie aplikácie. Taktiež v nej bude ukázané, ako nainštalovať a nakonfigurovať PostgreSQL databázu tak, aby bola aplikácia schopná vykresliť mapu a vygenerovať vyhľadávací graf z mapových podkladov uložených v tejto databáze.

Následne budú v tejto kapitole predstavené aplikácie a nástroje, ktoré sa používajú k spracovaniu a následnému importu dát z OSM projektu do PostgreSQL databáze. Inštalačné balíčky všetkých nástrojov a aplikácií sú priložené k tejto práci, ale pri každom bude uvedený odkaz, odkiaľ sa v čase vypracovanie tejto práce dal daný inštalačný balíček stiahnuť.

V prípade, že užívateľ využíva čisto zobrazovanie mapy pomocou tilingu a už vytvorené grafové dáta, PostgreSQL databáza nie je potrebná na beh aplikácie. Táto databáza sa využíva iba pri generovaní nového grafu a vykresľovaní mapy z databáze.

7.1 Užívateľské rozhranie aplikácie

Po spustení aplikácie sa otvorí okno s menu nachádzajúce sa na vrchnej lište. Toto menu má šesť záložiek: Graph, Map, Pathfinding, IncreaseZoom, DecreaseZoom a Settings.

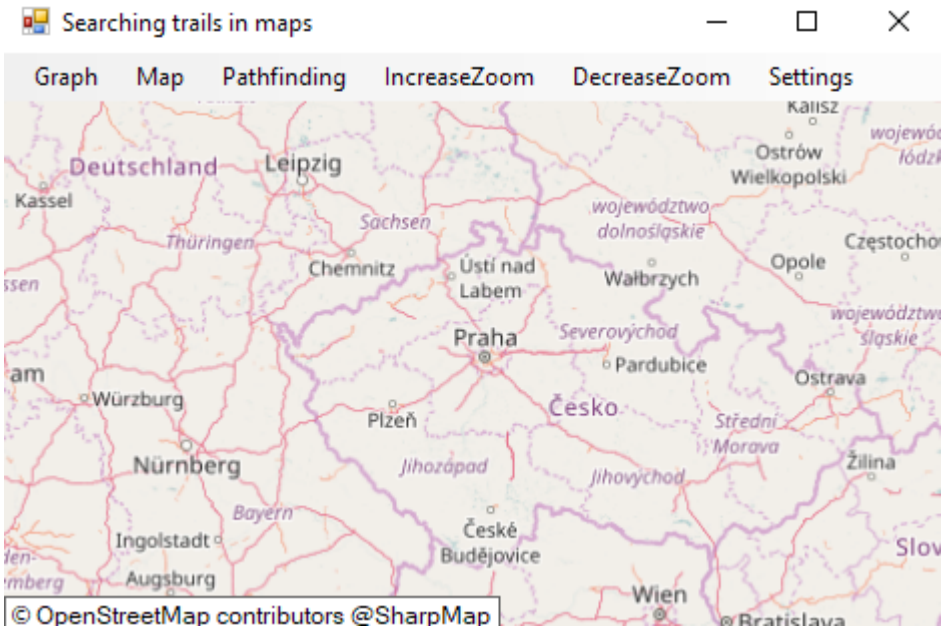
Tlačidlá IncreaseZoom a DecreaseZoom slúžia na priblíženie a oddialenie mapy. Navigácia po nej je v aplikácií riešená kliknutím na miesto na mape, na ktoré sa mapa vycentruje. Tento spôsob navigácie bol zvolený kvôli pomalému prekresľovaniu mapy pri použití vykresľovania z databáze.

Záložka Settings obsahuje štyri tlačidlá. Set connection, pomocou ktorého sa nastavuje databáza, s ktorou bude program pracovať a tlačidlo Set path to SRTM, pomocou ktorého sa dá nastaviť cesta k súborom s výškovými údajmi. Aplikácia si totiž nesťahuje výškové dáta sama. A to hlavne z dôvodu, že pri vytváraní kontúr program Phyghtmap stiahne a uloží tieto súbory do priečinka na disk. Set path to Themes a Set path to Symbols, ktorými sa nastavuje cesta k súborom potrebným na vykreslenie grafu.

Záložka Graph

Po kliknutí na záložku Graph sa zobrazia tlačidlá New a Load. New slúži na vytvorenie nového grafu z mapových údajov uložených v databáze. Po kliknutí na toto tlačidlo sa zobrazí dialóg na výber priečinka, do ktorého bude graf uložený. V prípade, že daný priečinok bude obsahovať súbory s názvami, ktoré sa využívajú na pomenovanie grafových súborov (nodes.gph, edges.gph a profile.gph), pôvodné súbory budú prepísané. Na vytvorenie grafu je potrebné nastaviť cestu k výškovým dátam pomocou Settings/Set Path To SRTM a taktiež je potrebné nastaviť správne reťazec, ktorým sa aplikácia pripojí na databázu. To sa robí pomocou Settings/Set connection.

Tlačidlo Load slúži na načítavanie už existujúceho grafu. Po kliknutí na túto záložku sa užívateľovi zobrazí dialógové okno, ktoré slúži na výber priečinka,



Obr. 7.1: Úvodná obrazovka aplikácie

v ktorom sa nachádzajú grafové súbory. V prípade, že sa z vybraného priečinka nepodarí graf načítať, vypíše sa na obrazovku správa.

Záložka Map

Záložka Map slúži k zobrazovaniu mapy. Po kliknutí na túto záložku sa rozbalia dve možnosti Render a Use tiling. Render sa pokúsi vykresliť mapu priamo z mapových dát uložených v databáze. Toto vykresľovanie je, hlavne pri väčšom oddialení mapy, výkonovo náročné a prekresliť obrázok pri posune po mape alebo pri zmene priblíženia môže trvať aj desiatky sekúnd. Na to aby sa dala táto možnosť využiť treba správne nastaviť Settings/Set connection, Settings/Set path to Themes a Settings/Set path to Symbols. Posledné dve spomenuté nastavujú cestu, ktorou sa aplikácia dostane k obrázkom potrebným na vykresľovanie.

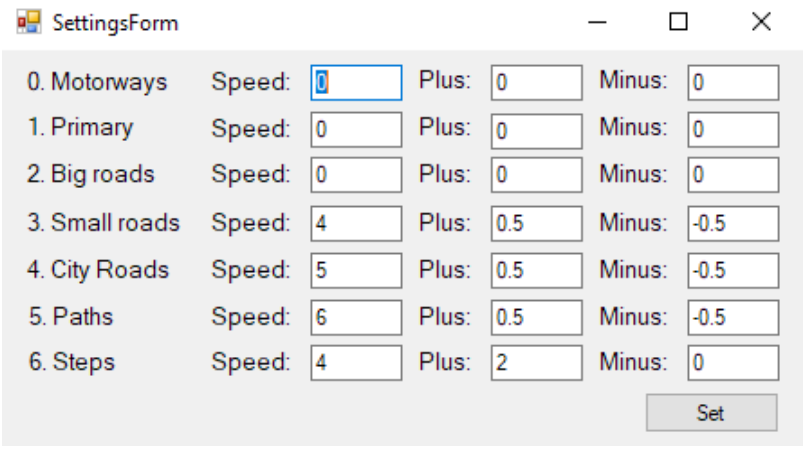
Kliknutím na tlačidlo Use tiling sa na zobrazenie mapy využije tiling stiahnutý z internetu, konkrétne zo serverov projektu OpenStreetMaps. Táto možnosť je zvolená automaticky pri spustení aplikácie. Zvolený framework neumožňuje vytvoriť tiling z vykreslenej mapy a pridanie tejto funkcionality je vysoko nad rozsah tejto práce.

Záložka Pathfinding

Záložka Pathfinding skrýva nástroje určené na vyhľadávanie najkratšej cesty na mape. Konkrétne tlačidlá Find a Settings, ktoré slúžia na nastavenie parametrov vyhľadávania. Okrem toho, záložka Pathfinding obsahuje tlačidlo Last Search Time, namiesto ktorého sa po prvom vyhľadávaní najkratšej cesty zobrazí čas, ktorý trvalo vyhľadať poslednú cestu.

Po kliknutí na tlačidlo Find sa zobrazí nové okno s názvom Find Shortest Path, skrátene FSP, ktoré obsahuje tri tlačidlá From, To a Find. Po kliknutí na tlačidlá From a To sa okno FSP schová až dokiaľ užívateľ kliknutím na mapu

nezvolí príslušný bod. V prípade, že užívateľ navolil začiatočný a koncový bod vyhľadávania sa po kliknutí na tlačidlo Find zatvorí okno FSP a vyhľadá sa a zakreslí najkratšia cesta z bodu From do bodu To na mapu.



Obr. 7.2: Okno slúžiace na nastavenie parametrov vyhľadávania.

Po kliknutí na tlačidlo Settings sa zobrazí nové okno určené na nastavenie parametrov vyhľadávania (obr. 7.2). Nachádza sa v ňom sedem riadkov, z ktorých každý reprezentuje iný typ cesty. Motorways sú diaľnice, Primary sú cesty prvej triedy, Big Roads reprezentujú cesty spájajúce mestá, Small roads reprezentujú ostatné cesty určené pre autá, City Roads sú cesty v rezidenčných štvrtiach, Paths sú turistické cesty a Steps sú schody. Pri každom type sa dá navoliť rýchlosť pohybu (speed), penalta za každý plusový meter (Plus) a za každý mínusový meter (Minus). Rýchlosť nemôže byť záporná. Vzorec použitý na výpočet ohodnotenia hrany v aplikácií je nasledovný: dĺžka hrany $\div$ rýchlosť + plusové výškové metre $*$ Plus + mínusové výškové metre $*$ Mínus. V prípade, že tento vzorec je záporný berie sa nula.

7.2 Príprava mapových dát

V prípade, že chce užívateľ využívať možnosti vykreslenia mapy a vytvorenia grafu je potrebné, aby mala aplikácia prístup k PosgreSQL databáze, do ktorej sú importované mapové dáta. Pretože generovanie a vykresľovanie grafu využíva veľké množstvo dát z tejto databáze, odporúča sa, aby bežala na rovnakom počítači ako beží aplikácia.

Samotný import dát do databáze sa dá urobiť buď načítaním exportu z existujúcej databáze, alebo pomocou programu Osm2pgsql. Odporúčaný postup je použiť export z databáze, pretože druhý spomínaný je časovo náročný a vyžaduje inštaláciu viacerých nástrojov a knižníc, ktoré tieto nástroje potrebujú.

Práca s príkazovým riadkom

Pretože nástroje, ktoré budú predstavené v nasledujúcej časti tejto kapitoly, bežia na príkazovom riadku, bude v tejto sekcii ukázaná základná navigácia súborovým systémom a spúšťanie súborov pomocou príkazového riadku platformy Windows.

Ak je potrebné dostať sa do priečinku D:\AAA\BBB, použijú sa na to nasledovné dva príkazy:

```
D:
cd D:\AAA\BBB
```

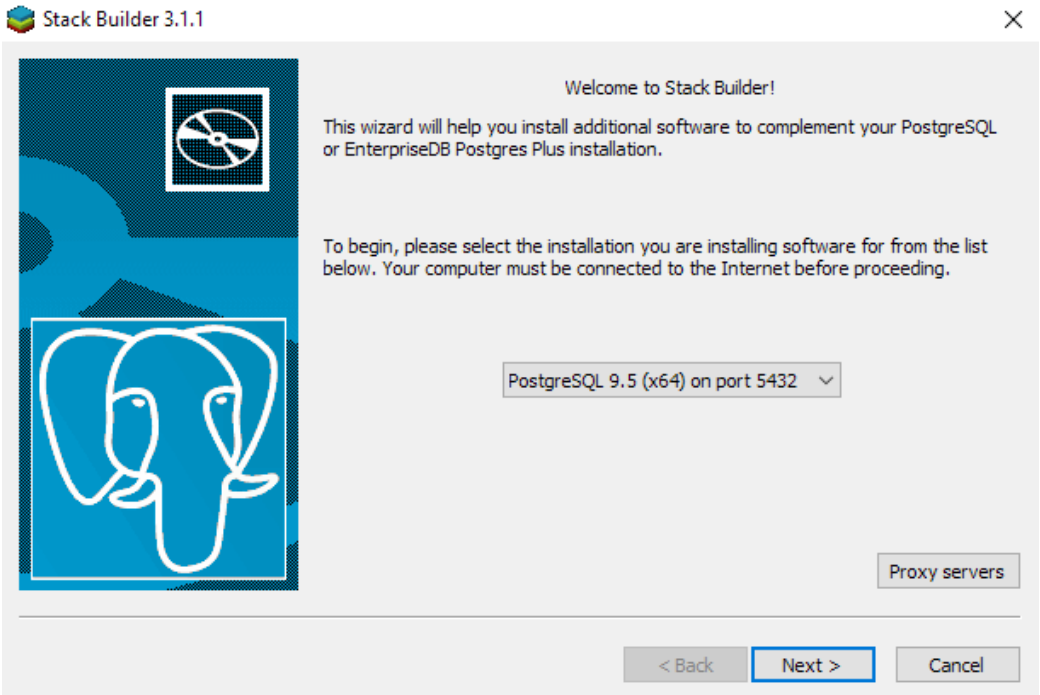
Spustenie súboru D:\AAA\BBB\file.exe sa urobí nasledovne:

```
D:
cd D:\AAA\BBB
file.exe
```

Inštalácia PostgreSQL a vytvorenie databázy

Inštalácia PostgreSQL sa robí pomocou inštalačného programu, ktorý sa dá stiahnuť priamo zo stránky projektu.

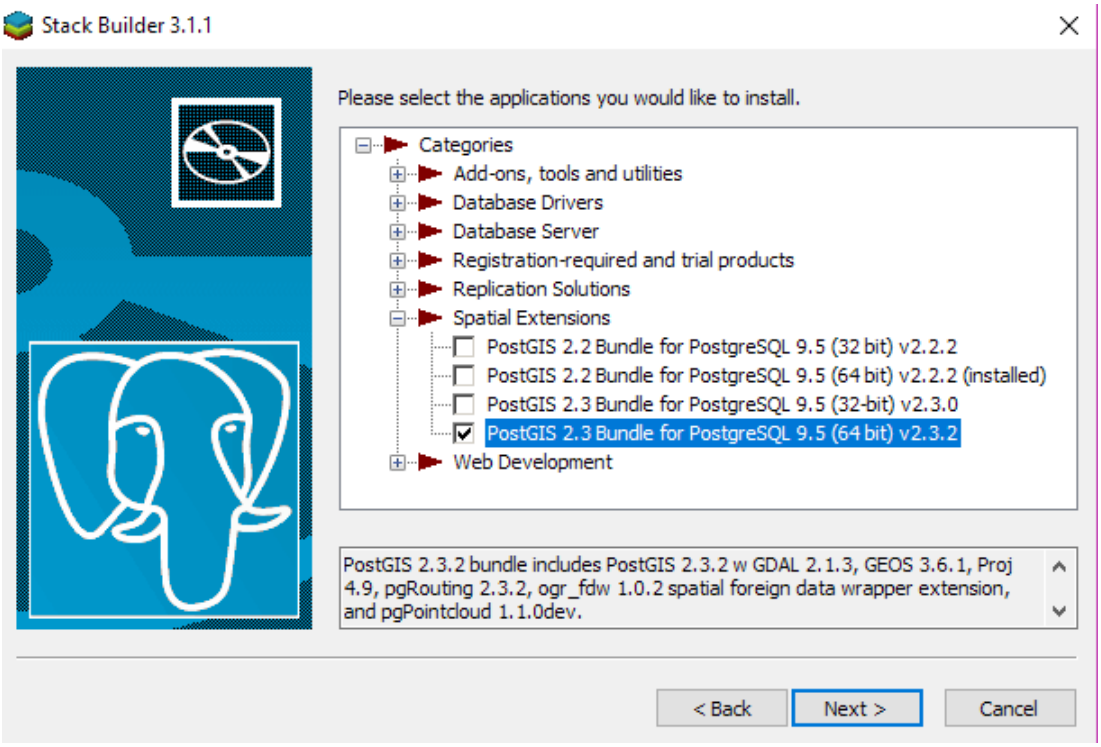
Po inštalácií PostgreSQL je potrebné spustiť aplikáciu Stack Builder, ktorá sa nainštalovala spoločne s PostgreSQL, a nainštalovať jej rozšírenie PostGIS. Samotná inštalácia je veľmi jednoduchá, na úvodnej stránke si stačí vybrať správnu databázu (obr. 7.3). Následne si užívateľ vyberie z možných rozšírení PostGis (obr. 7.4).



Obr. 7.3: Inštalácia PostGisu.

Načítanie exportu databáze sa urobí nasledovne: Príkazovým riadkom sa dostaneme do adresára bin v inštalačnom priečinku PostgreSQL a následne sa spustí príkaz:

```
psql -h <Názov počítača, na ktorom beží databáza>
-U <Užívateľ databázy> -W -d <Názov existujúcej databázy>
-f <Cesta k exportu>
```



Obr. 7.4: Inštalácia PostGisu.

Tento príkaz potrebuje aby databáza do ktorej budú dáta exportované existovala. V prípade, že chce užívateľ vytvoriť novú databázu, je odporučené použiť aplikáciu pgAdmin III. Vytvorenie databázy pomocou tejto aplikácie je veľmi jednoduché:

V prvok kroku sa vyberie databázový server, pod ktorým užívateľ chce vytvoriť novú databázu. Pravým tlačidlom myši sa klikne na položku databases a vyberie sa New Database... (obr. 7.5). Následne stačí vyplniť názov a vlastníka databáze, a nakoniec na vytvorenie potvrdiť tlačidlo OK, ako je ukázané na obrázku 7.6.

Teraz už len stačí pridať do novo vytvorenej databázy rozšírenie pre prácu s viacrozmernými dátami. To sa urobí pomocou okna Execute arbitrary SQL queries (obr. 7.7), v ktorom sa spustí nasledujúci príkaz:

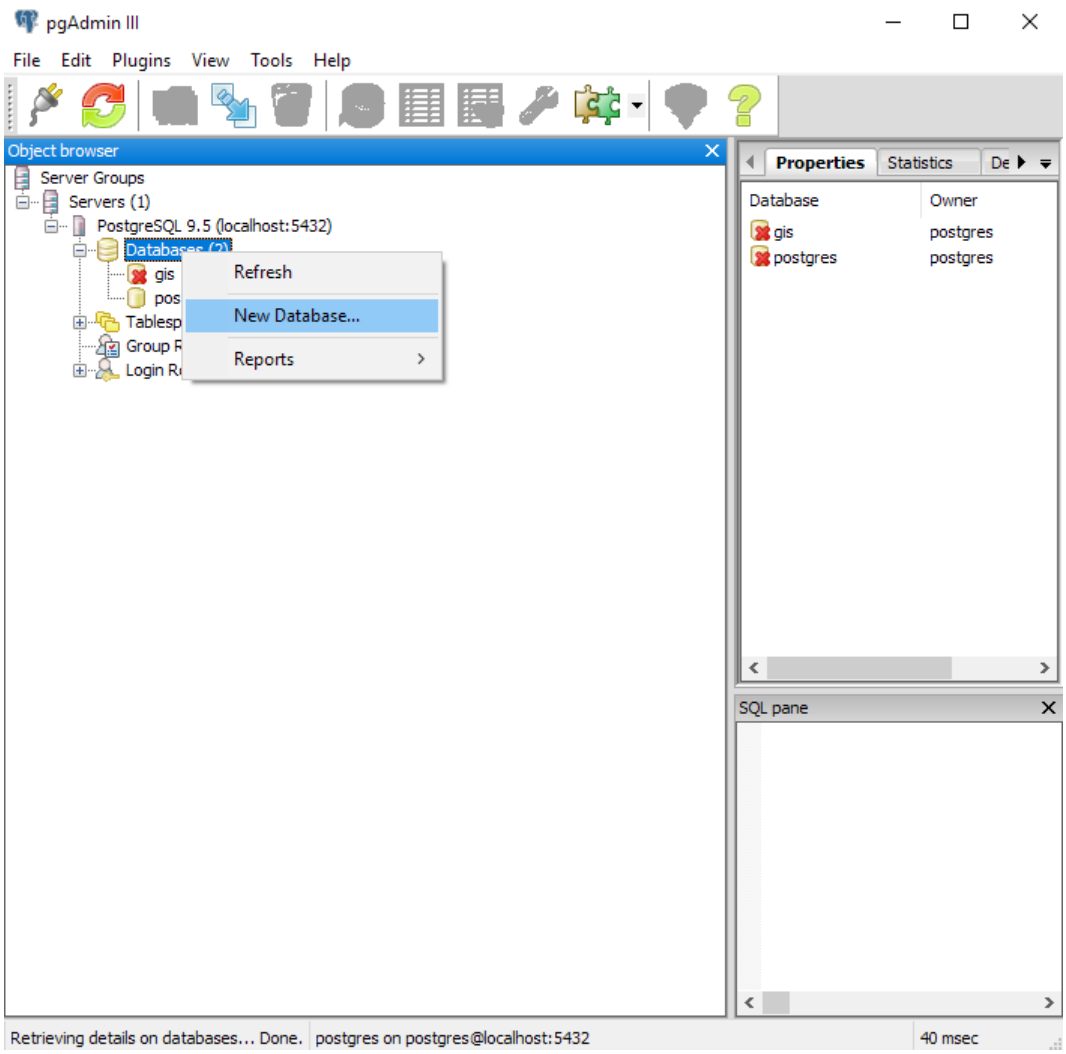
```
CREATE EXTENSION postgis
```

Import OSM dát do databázy

Na stránke http://wiki.openstreetmap.org/wiki/Planet.osm#Downloading sú všetky úložiská, z ktorých sa dajú OSM dáta získať. V tejto práci bola využitá stránka https://download.geofabrik.de/, na ktorej sa dá vybrať, ktorý kontinent alebo krajina sa má stiahnuť.

Po získaní dát je si potrebné nainštalovať program Osm2pgsql. Inštalácia tohto programu je jednoduchá, stačí len rozbaliť inštalačný balíček do priečinku.

Program Osm2Pgsql pri spustení potrebuje nastaviť správne prepínače aby skutočne spravil to čo má.



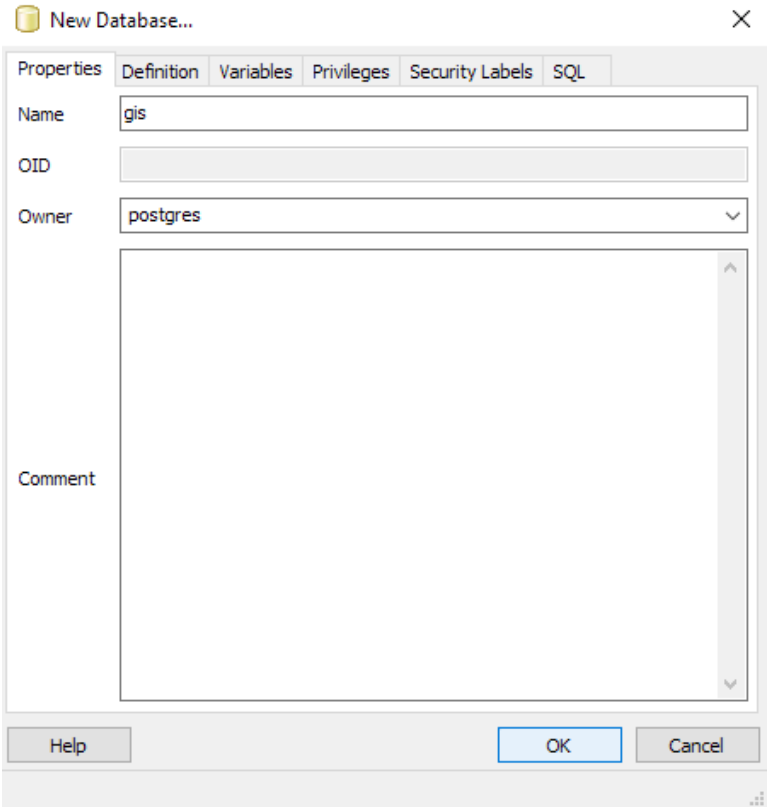
Obr. 7.5: Vytvorenie novej datázy.

```
osm2pgsql.exe -d <Názov databázy> -U <Užívateľ databázy> -W
-H <Názov počítača na ktorom beží databáza> -P <port>
-C <Veľkosť cache pamäte> --slim -S <Cesta k štýlu>
<Cesta k OSM súboru>
```

Príklad príkazu pre štandartné nastavenia:

```
osm2pgsql.exe -d gis -U postgres -W -H localhost -P 5432
-C 2000 --slim -S D:\cygwin-package\default.style
D:\czech_republic.osm
```

Je potrebné si dať pozor, pretože štýl, ktorý využíva táto aplikácia, sa od líši štandardného štýlu programu tým, že má pridané riadky, ktoré slúžia na import kontúr do databáze. V inštalačnom balíčku priloženom k tejto práci sa nachádza už upravený štýl.



Obr. 7.6: Vytvorenie novej datázy.

```
        node,way    contour         text        linear
        node,way    contour_ext     text        linear
        node,way    ele             text        linear
```

V prípade, že je potrebné pridať ďalšie dáta do už existujúcej databáze je nutné použiť prepínač -a, čo užívateľ využije v prípade, že bude chcieť importované dáta doplniť o kontúry. Štandardný príkaz na pridanie kontúr bude teda vyzerať nasledovne:

```
        osm2pgsql.exe -a -d gis -U postgres -W -H localhost -P 5432
        -C 2000 --slim -S D:\cygwin-package\default.style
        D:\srtm.osm
```

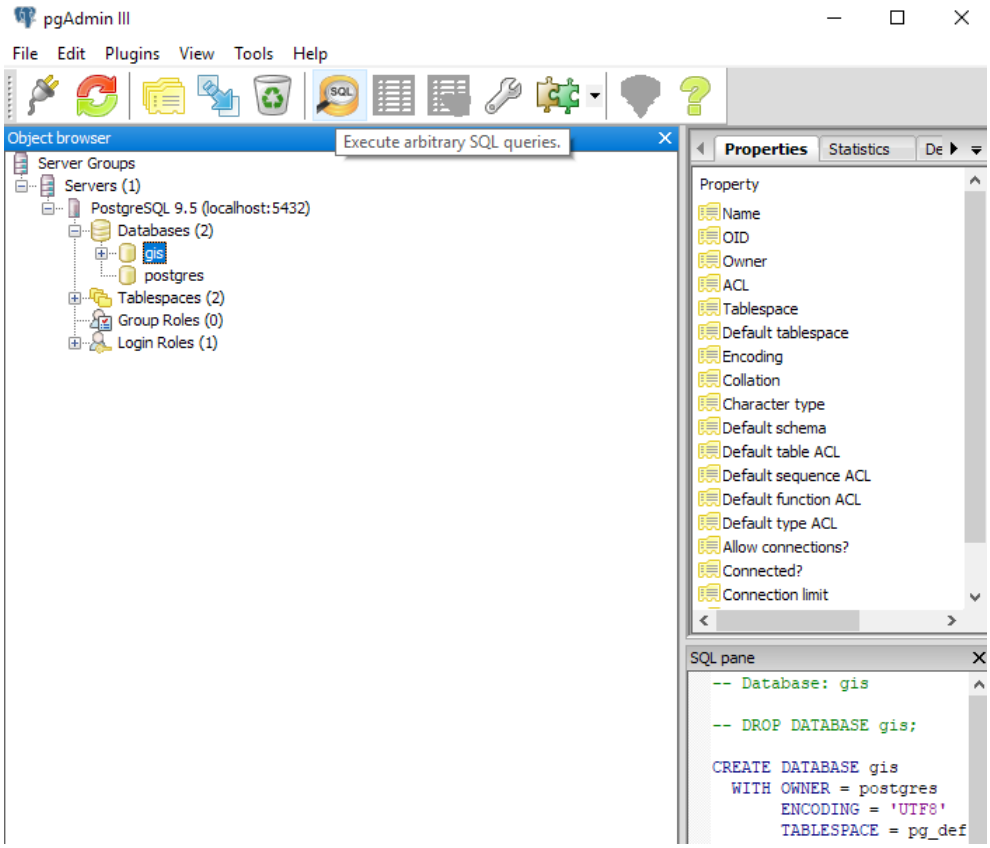
Generovanie kontúr

Na vytvorenie kontúr, ktoré bude možné importovať do databázy, bol v tejto práci použitý program phyghtmap.

Inštalácia tohto programu vyžaduje nainštalovanie nasledujúcich programov v tomto poradí:

- python[1]
- python-setuptools[2]

[1] https://www.python.org/downloads/
[2] https://sourceforge.net/projects/matplotlib/files/matplotlib/



Obr. 7.7: Otvorenie okna Execute arbitrary SQL queries.

- python-matplotlib[3]
- python-numpy[4]
- python-beautifulsoup[5]
- phyghtmap[6]

Prvé štyri sa inštalujú pomocou klasického inštalačného programu. Python-beautifulsoup je potrebné rozbaliť do priečinku, v ktorom je nainštalovaný Python, napríklad do c:\python27\BeautifulSoup-3.2.0. Rovnako je nutné rozbaliť aj phyghtmap. Následne je potrebné spustiť nasledujúci príkaz v oboch priečinkoch:

```
c:\python27\python.exe setup.py install
```

V tomto momente je už možné začať program phyghtmap používať. Na vygenerovanie kontúr v tejto práci bol použitý príkaz:

```
phyghtmap.exe -a 11:48:20:52 -s 10 --max-nodes-per-way=0
--max-nodes-per-tile=0 --start-node-id=10000000000
--start-way-id=10000000000
```

[3]https://pypi.python.org/pypi/setuptools#windows
[4]https://sourceforge.net/projects/numpy/files/NumPy/
[5]http://pypi.python.org/packages/source/B/BeautifulSoup/BeautifulSoup-3.2.0.tar.gz#md5=9c0f5d246ecfcf5f0905a47b1466e140
[6]http://katze.tfiu.de/projects/phyghtmap/phyghtmap_1.42.orig.tar.gz

Prepínačom -a sa špecifikuje aké kontúry daného územia budú vygenerované. Prepínače –max-nodes-per-way a –max-nodes-per-tile slúžia na rozdeľovanie kontúr do viacerých súborov. Nastavenie oboch na 0 znamená, že vo výsledku sa vytvorí jeden veľký súbor. Prepínače –start-node-id a –start-way-id nastavujú, od akej počiatočnej hodnoty začnú ID vygenerovaných vrcholov a ciest. Je potrebné zvoliť dostatočne veľké číslo, aby nenastali kolízie s ID objektov v OSM dátach. Prepínač -s slúži k nastaveniu veľkosti kroku v metroch pri generovaní kontúr.

8. Programátorská dokumentácia

Aplikácia sa dá rozdeliť na štyri hlavné časti:

- Užívateľské rozhranie.
- Vykresľovanie mapy.
- Vytváranie grafu.
- Vyhľadávanie najkratšej cesty.

8.1 Užívateľské rozhranie

Užívateľské rozhranie tejto aplikácie je veľmi jednoduché, pretože sa táto aplikácia skôr zameriava na výber a ukážku implementácie konkrétnych algoritmov a dátových štruktúr na vyhľadávanie najkratšej cesty, viac než na každodenné použitie.

Samotné rozhranie je vytvorené ako WindowsFormApplication, a slúži na komunikáciu medzi užívateľom a ostatnými časťami aplikácie. Z programátorského hľadiska sa tu nenachádza nič komplikované, čo by stálo za zmienku.

8.2 Vykresľovanie mapy

V užívateľskom rozhraní sa nachádza Mapbox, ktorý sa stará o vykreslenie mapy na obrazovku. Predtým ako môžeme mapu vykresliť je potrebné nastaviť odkiaľ má Mapbox brať dáta a ako ich následne má zakresliť. To sa deje pomocou tried, ktoré implementujú interface ILayer. Pretože týchto tried je na vykreslenie vlastnej mapy potrebných veľké množstvo, aplikácia využíva na ich vytváranie a predávanie Mapboxu triedu LayerFactory.

LayerFactory

LayerFactory slúži na vytváranie inštancií tried implementujúcich interface ILayer, a na ich následné posielanie v správnom poradí tomu, kto si o nich požiada. Interne táto trieda obsahuje pole typu ILayer, kam počas inicializácie dosadí jednotlivé inštancie tohto typu, pomocou metód CreateOSMVectorLayer, CreateOSMLabelLayer a CreateShapeLayer. Jednotlivé inštancie si potom externé triedy vedia získať pomocou property HasNext a metódy GetNext.

Druhá dôležité funkcia tejto LayerFactory je naplnenie statických inštancií triedy Dictionary, ktoré sa používajú pri samotnom vykresľovaní mapy.

CreateOSMVectorLayer

CreateOSMVectorLayer, ako už názov napovedá, slúži k vytváraniu inštancií VectorLayer z inštancie triedy ktorá implementuje rozhranie IOSMLayer. OSM je v názve metódy z dôvodu, že sú pri vytváraní inštancií očakávané dáta v OSM formáte, uložené v konkrétnej tabuľke PostgreSQL databázy.

Pred tým než sa vytvorí samotný VectorLayer, metóda skontroluje či je vytvorená tabuľka z ktorej plánuje čerpať dáta, a ak nie tak túto tabuľku vytvorí pomocou IDatabaseAccess. Následne vytvorí priestorový index nad stĺpcom way, ktorý je štandardne stĺpec obsahujúci informácie o geometrii jednotlivých riadkov tabuľky.

CreateOSMLabelLayer

Metóda CreateOSMLabelLayer je veľmi podobná metóde CreateOSMVectorLayer, s tým rozdielom, že sa nevytvorí VectorLayer, ale LabelLayer, ktorý slúži na vykresľovanie textu.

CreateShapeLayer

Posledná z metód na generovanie vrstiev mapy je CreateShapeLayer, ktorá vytvorí VectorLayer so zdrojom dát vo forme shapefilu.

8.3 Vytváranie grafu

Dáta potrebné na tvorbu grafu sú získavané z tabuliek vytvorených aplikáciou v PostgreSQL. Prvá z nich obsahuje všetky dopravné a pešie cesty vo forme ID cesty, príznak highway ktorý nesie informáciu o type cesty a usporiadaný zoznam vrcholov ktoré danú cestu tvoria. Druhá tabuľka obsahuje všetky vrcholy vo forme ID vrcholu, lat a lon, kde lat a lon sú súradnice bodu v súradnicovom systéme ESPG 3857[1] krát sto, aby bolo možné ich uložiť ako dátový typ integer do databázy.

Dáta z tabuliek je potrebné získavať po častiach, pretože napríklad tabuľka pre vrcholy pre Českú republiky má viac než 113 miliónov riadkov.

Samotný graf je vytváraný v troch fázach:

- Predspracovanie ciest.
- Spracovanie vrcholov.
- Spracovanie ciest.

Niektoré body sa v OSM dátach používajú k opisu tvaru cesty. Tento typ bodov nie je vhodné mať v grafe pomocou ktorého sa vyhľadáva najkratšia cesta, pretože výslednú cestu nezmenia a výrazne by zväčšili veľkosť grafu.

V prvej fáze sú prejdené všetky cesty a označené začiatočné, koncové, a tie body, ktoré patria aspoň dvom rôznym cestám, t.j. tvoria križovatku. Tieto body budú vrcholmi vo výslednom grafe, a teda v zvyšku tejto kapitoly bude pod označením vrchol myslený práve jeden z týchto bodov. Hranou grafu bude myslená časť cesty, ktorá začína a končí vrcholmi a neobsahuje ďalšie vrcholy. V prípade, že hrana obsahuje body, ktoré nie sú vrcholmi bude pre tieto body použité označenie profil hrany.

Navyše pri v prvej fáze sú pôvodné OSM ID vrcholov prečíslované tak aby boli očíslované od 0 do ich počtu - 1.

[1] https://epsg.io/3857

V druhej fáze vytvárania grafu sa vytvorí súbor obsahujúci informácie o jednotlivých vrcholoch. Na vytvorenie tohto súboru je použitá trieda implementujúca rozhranie INodeAccess, konkrétne trieda DefaultNodeAccess.

V tretej a finálnej fáze aplikácia znova prejde všetky cesty a tentoraz začne vytvárať jednotlivé hrany. V každej ceste nájde dvojice rôznych vrcholov, medzi ktorými neleží žiaden iný vrchol. Pre každú z dvojíc vrcholov je potrebné vytvoriť hranu, k čomu slúži rozhranie IEdgeAccess a v tejto práci kokrétne trieda DefaultEdgeAccess.

V mapových dátach z OSM projektu existujú cesty typu ...,v_i,b_k,...,b_n,v_i,..., kde v_i je vrchol, a b_k,...b_n sú body. Takáto hrana, nazývaná smyčka smeruje sama do seba a nijak neovplyvní vyhľadávanie najkratšej cesty, preto je ignorovaná pri vytváraní hrán grafu.

Hrana sa vytvára tak, že aplikácia najprv skontroluje či prvý vrchol je menší ako ten druhý a ak nie bude spracovávať opačnú hranu. Po kontrole orientácie hrany sa uložia ID vrcholov v správnom poradí, nastavia sa odkazy na ďalšie hrany na prázdnu hodnotu a následne sa zapojí táto vytváraná hrana do zoznamu susedov oboch vrcholov. Zapojenie do zoznamu vrcholov prebieha tak, že sa najskôr sa nájde posledná už pridaná hrana zo zoznamu vrcholov, pre daný vrchol, a novovytváraná hrana sa pripojí za ňu do zoznamu. V prípade, že po pripojení novej hrany má neprázdnu hodnotu pre ďalšiu hranu pre oba vrcholy, vymažeme túto hranu zo zoznamu hrán na dokončenie. Zároveň do tohto zoznamu pridáme novovytvorenú hranu. Teraz už treba spočítať a uložiť súčet vzdialeností a plusových a mínusových metrov medzi jednotlivými bodmi cesty. Nakoniec je potrebné ešte uložiť profil cesty v správnom poradí pomocou rozhrania IProfileAccess do súboru.

Po spracovaní všetkých ciest je graf vytvorený a pripravený na používanie.

DefaultNodeAccess

DefaultNodeAccess slúži na čítanie a ukladanie informácií o vrchole. Pri práci načíta celý súbor s informáciami o vrcholoch do RAM pamäte. Súbor, v ktorom si táto trieda ukladá informácie je binárny súbor, ktorý začína hlavičkou a po nej nasleduje postupnosť trojíc 32 bitových integerov, kde prvá trojica sú informácie o vrchole s ID 0, druhá o vrchole s ID 1, atď. Vďaka prečíslovaniu ktoré bolo urobené v prvej fáze vytvárania grafu, tento súbor bude súvislý. Pre každý vrchol sa ukladá informácia lat, lon a ID hrany, ktorá ho obsahuje.

DefaultEdgeAccess

DefaultEdgeAccess slúži k čítaniu a ukladaniu informácií o hrane grafu. Rovnako ako pri vrcholoch, aj hrany sú pri vytváraní prečíslované a teda je možné použiť rovnaký spôsob prístupu k informáciám o hrane s konkrétnym ID ako pri vrcholoch. Samotná hrana je komplikovanejšia dátová štruktúra ako vrchol. Každá hrana postupnosť prvkov A, B, NextA, NextB, ProfileOffset, ProfileLenght, Flag, Plus, Minus, Distance, kde:

- A a B sú ID vrcholov, ktoré tvoria hranu pričom platí, že A je menšie ako B a ostatné informácie o hrane sú ukladané v orientácii hrany z A do B.

- NextA a NextB sú ID ďalších hrán v zozname susedov konkrétneho vrcholu hrany. NextA ukazuje na ďalšiu hranu, ktorá obsahuje A, a rovnako nextB ukazuje na hranu, ktorá obsahuje vrchol B.
- ProfileOffset a ProfileLenght, slúžia k tomu aby sa dalo pristúpiť informáciám o profile cesty.
- Flag slúži k uchovaniu informácií o vlastnostiach hrany.
- Plus a Minus sú informácie o plusových a mínusových výškových metroch danej hrany.
- Distance je dĺžka hrany v metroch.

DefaultProfileAccess

DefaultProfileAccess slúži k pristupovaniu k informáciám o profile danej cesty. Je to veľmi jednoduchý binárny súbor, ktorý je tvorený dvojicami za sebou idúcich 32 bitových celých čísel, ktoré reprezentujú lat a lon konkrétnych bodov. Na prístup k profilu konkrétnej hrany, je potrebné vedieť na ktorom offsete tento profil začína, a akú má dĺžku.

8.4 Vyhľadávanie najkratšej cesty

O vyhľadávanie najkratšej cesty medzi dvoma bodmi sa v aplikácií stará rozhranie IShortestPathFinder, a konkrétne trieda AStar, ktorá toto rozhranie implementuje.

Na ukladanie informácií o vrchole do prioritnej fronty sa využíva privátna trieda PriorityNode, ktorá vlastne reprezentuje dvojicu priorita a ID vrcholu, kde pod pojmom priorita myslí F-skóre vrcholu.

AStar

Trieda AStar je implementácia algoritmu A*. Na množina uzavretých vrcholov je implementovaná pomocu HashSetu a množina otvorených vrcholov je implementovaná pomocov rozhrania IPriorityHeap, konkrétne triedou BinaryHeap. Na ukladanie informácií o G-Skóre a o hrane, ktorou sa prišlo do konkrétneho vrcholu sa využíva Dictionary.

BinaryHeap

Trieda BinaryHeap je implementácia binárnej haldy, ktorá navyše ešte využíva Dictionary na uchovávanie informácií o tom, ktoré vrcholy sa nachádzajú vo fronte a na ktorej pozícií. Vďaka tomuto prístupu sa vyhľadávanie zrýchlilo viac než dvojnásobne, oproti implementácií ktorá nevyužívala takúto triedu.

Záver

Výsledkom práce je aplikácia, ktorá dokáže vyhľadávať najkratšie cesty v reálnych mapových podkladoch a zároveň berie do úvahy výškové dáta. Vyhľadávanie je rýchle, cesty o dĺžke 100km zvládne aplikácia nájsť v čase menšom ako 1 sekunda, a to dokonca aj pri nastavení, ktoré sa úplne vyhýba diaľniciam a rýchlostným cestám. Navyše sa vyhľadávanie dá prispôsobiť potrebám užívateľa bez nutnosti znova vytvoriť graf, čo túto aplikáciu odlišuje od väčšiny mapových aplikácií, ktoré povoľujú užívateľovi iba vybrať si jeden z predpripravených profilov vyhľadávania.

Okrem vyhľadávania aplikácia umožňuje vygenerovať graf z mapových dát v OSM formáte, ktorý je veľmi kompaktný a zároveň umožňuje rýchly prístup k informáciám potrebným na hľadanie najkratšej cesty. Veľkosť grafových súborov pre mapu Českej republiky je 124MB, zatiaľ čo veľkosť pôvodných mapových dát v OSM XML formáte je 14.5GB. Doba trvania generovania grafu pre mapy Českej republiky je v jednotkách až desiatkach minút.

Aplikácia navyše dokáže aj vykresliť mapové dáta, s vlastným štýlom vykresľovania a kontúrami. Kvôli chýbajúcej podpore vytvárania tilingu z vykreslenej mapy vo frameworku použitom v tejto aplikácií je zobrazovanie vykresľovanej mapy pomalé. Pridanie podpory pre vytváranie tilingu je nad rozsah tejto práce, ale aplikácia čiastočne rieši problém tým, že umožňuje užívateľovi zvoliť si medzi mapou vykreslenou z mapových dát alebo mapou zobrazenou zo serverov projektu OpenStreetMaps.

Na spustenie aplikácie na platforme Windows je potrebná verzia 4.5.2 .NET frameworku. Po spustení je aplikácia schopná načítať existujúci graf a v prípade, že daný počítač má pripojenie do internetu aj OpenStreetMaps tiling. Generovanie a vykresľovanie grafu z mapových dát vyžaduje pripojenie na PostgreSQL databázu, v ktorej sú tieto dáta uložené v správnom formáte.

Zoznam použitej literatúry

de Ferranti, J. ASTER Digital Elevation Data. URL `http://www.viewfinderpanoramas.org/reviews.html#aster`.

Dibbelt, J., Strasser, B. a Wagner, D. (2015). Customizable Contraction Hierarchies. *Journal of Experimental Algorithmics. ACM.*, **21**, 1.5:1—1.5:49.

Dijkstra, Edsger, W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, **1**, 269–271.

ESRI (1998). ESRI Shapefile Technical Description.

Foundation, F. S. (1991). GNU General Public Licence version 2.0. URL `https://opensource.org/licenses/gpl-2.0.php`.

Foundation, F. S. (2007). GNU lesser General Public Licence version 3.0. URL `https://www.gnu.org/licenses/lgpl-3.0.en.html`.

Foundation, T. A. S. (2017). Apache License, Version 2.0. URL `https://www.apache.org/licenses/LICENSE-2.0`.

Fredman, M. L. a Tarjan, R. E. (1987). Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the Association for Computing Machinery*, **34**(3), 596–615.

Google (2013). A Brief History of Google Maps. URL `https://web.archive.org/web/20130605102259/https://maps.google.com/help/maps/helloworld/tips/history.html/`.

Hart, P. E., Nilsson, N. J. a Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics SSC4*, **4**(2), 100–107.

Hirt, C., Filmer, M. S. a Featherstone, W. E. (2010). Comparison and validation of recent freely-available ASTER-GDEM ver1, SRTM ver4.1 and GEODATA DEM-9S ver3 digital elevation models over Australia. *Australian Journal of Earth Sciences*, **57**(3), 337–3474.

International, E. (2013). Standard ECMA-404.

International, E. (2016). Standard ECMA-262.

Matoušek, J. a Nešetřil, J. (2009). *Kapitoly z diskrétní matematiky*. Čtvrté vydání. Univerzita Karlova v Praze, Nakladatelství Karolinum, Praha. ISBN 978-80-246-1740-4.

Microsoft (2017a). C# Reference. URL `https://docs.microsoft.com/en-us/dotnet/articles/csharp/language-reference/index`.

Microsoft (2017b). .NET Documentation. URL `https://docs.microsoft.com/en-us/dotnet/articles/csharp/language-reference/index`.

NASA (2017). SRTM.

Oracle (2017). Java Language and Virtual Machine Specifications. URL https://docs.oracle.com/javase/specs/.

osm2pgsql (2017). osm2pgsql. URL https://github.com/openstreetmap/osm2pgsql.

Osmosis (2016). Osmosis project. URL https://github.com/openstreetmap/osmosis.

PostgreSQL (2017). The PostgreSQL Licence. URL https://www.postgresql.org/about/licence/.

Reese, B. (1999). AlphA*: An epsilon-admissible heuristic search algorithm.

Russell, S. a Norvig, P. (2002). Artificial Intelligence: A Modern Approach. *P*.

Saunders, S. (1999). A Comparison of Data Structures for Dijkstra's Single Source Shortest Path Algorithm.

SharpMap (2017). SharpMap. URL https://github.com/SharpMap.

Standard C++ Foundation (2017). Standard C++. URL https://isocpp.org/.

Thorup, M. (1999). Undirected single-source shortest paths with positive integer weights in linear time. *Journal of the ACM (JACM)*, **46**, 362–394.

Turner, L. (2011). Variants of the Shortest Path Problem. *Algorithmic Operations Research*, **6**(2), 91–104.

W3C (2017). Extensible Markup Language (XML). URL https://isocpp.org/.

Zoznam obrázkov

Prílohy

Príloha A

Obsah priloženého CD je nasledovný:

1. **Application Files** – Priečinok, ktorý obsahuje dáta potrebné na inštaláciu aplikácie.
2. **Graf** – Priečinok graf obsahuje súbory edges.gph, nodes.gph a profile.gph, ktoré reprezentujú vyhľadávací graf pre Českú republiku.
3. **Install** – Priečinok obsahujúci inštalačné balíky všetkých nástrojov potrebných na import dát z OSM XML formátu do PostgreSQL databáze.
4. **PathlessRouting** – Priečinok obsahujúci celú aplikáciu ako projekt určený pre Visual Studio.
5. **SRTM** – Priečinok obsahujúci výškové dáta, ktoré sa využívajú na tvorbu grafu.
6. **Symbols** – Priečinok obsahujúci obrázky, ktoré sa využívajú pri vykresľovaní mapy.
7. **Themes** – Priečinok obsahujúci obrázky, ktoré sa využívajú pri vykresľovaní mapy.
8. **PathlessRouting.application** – Súbor potrebný na inštaláciu aplikácie.
9. **setup.exe** – Súbor, ktorým sa nainštaluje aplikácia.